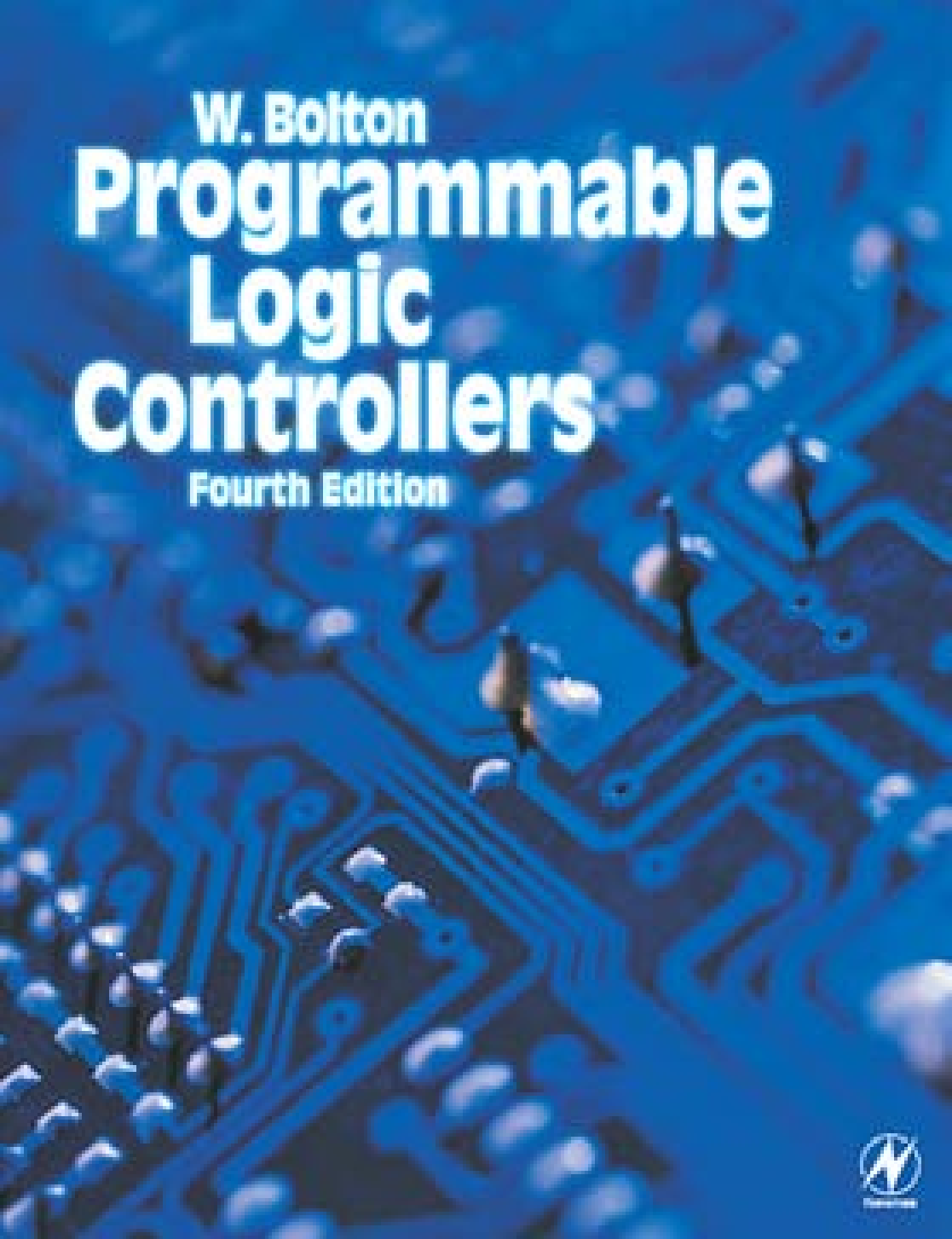

W. Bolton
Programmable Logic Controllers
Fourth Edition

VISIT…

Programmable Logic Controllers

Programmable Logic Controllers

Fourth Edition

W. Bolton

AMSTERDAM • BOSTON • HEIDELBERG • LONDON • NEW YORK • OXFORD
PARIS • SAN DIEGO • SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

ELSEVIER Newnes is an imprint of Elsevier Newnes

Newnes is an imprint of Elsevier
Linacre House, Jordan Hill, Oxford OX2 8DP
30 Corporate Drive, Suite 400, Burlington, MA 01803

First edition 1996
Second edition 2000
Third edition 2003
Fourth edition 2006

Copyright © 2006, W. Bolton. Published by Elsevier Newnes. All rights reserved

The right of W. Bolton to be identified as the author of this work has been asserted in accordance with the Copyright, Designs and Patents Act 1988

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means electronic, mechanical, photocopying, recordinh or otherwise without the prior permission of the publisher

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK; phone: (+44) 1865 843830; fax: (+44) (0) 1865 853333; e-mail: permissions@elsevier.co.uk. Alternatively you can submit your request on-line by visiting the Elsevier web site at http:www.elsevier.com/locate/permissions, and selecting *Obtaining permission to use Elsevier material*

Notice
No responsibility is assumed by the publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herin. Because of rapid advances in the medical sciences, in particular, independent verification of diganoses and drug dosages should be made

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Cataloging -in-Publication Data
A catalog record for this book is available from the Library of Congress

ISBN-13: 978-0-7506-8112-4
ISBN-10: 0-7506-8112-8

For information on all Newnes publications visit our website at www.newnespress.com

Printed and bound in the UK
06 07 08 09 10 10 9 8 7 6 5 4 3 2 1

Working together to grow libraries in developing countries
www.elsevier.com | www.bookaid.org | www.sabre.org
ELSEVIER BOOK AID International Sabre Foundation

Contents

			Preface	vii
1	**Programmable logic controllers**	1.1	Controllers	1
		1.2	Hardware	4
		1.3	Internal architecture	5
		1.4	PLC systems	10
			Problems	15
2	**Input-output devices**	2.1	Input devices	17
		2.2	Output devices	30
		2.3	Examples of applications	39
			Problems	41
3	**Number systems**	3.1	The binary system	44
		3.2	Octal and hexadecimal	45
		3.3	Binary arithmetic	47
		3.4	PLC data	51
			Problems	52
4	**I/O processing**	4.1	Input/output units	53
		4.2	Signal conditioning	59
		4.3	Remote connections	62
		4.4	Networks	69
		4.5	Processing inputs	75
		4.6	I/O addresses	76
			Problems	77
5	**Ladder and functional block programming**	5.1	Ladder diagrams	80
		5.2	Logic functions	84
		5.3	Latching	90
		5.4	Multiple outputs	91
		5.5	Entering programs	93
		5.6	Function blocks	94
		5.7	Program examples	100
			Problems	103
6	**IL, SFC and ST programming methods**	6.1	Intruction lists	108
		6.2	Sequential function charts	115
		6.3	Structured text	120
			Problems	124
7	**Internal relays**	7.1	Internal relays	132
		7.2	Ladder programs	133
		7.3	Battery-backed relays	136

	7.4	One-shot operation	137
	7.5	Set and reset	138
	7.6	Master control relay	142
		Problems	146
8 Jump and call	8.1	Jump	154
	8.2	Subroutines	156
		Problems	157
9 Timers	9.1	Types of timers	159
	9.2	Programming timers	160
	9.3	Off-delay timers	163
	9.4	Pulse timers	165
	9.5	Programming examples	166
		Problems	167
10 Counters	10.1	Forms of counter	173
	10.2	Programming	174
	10.3	Up and down counting	178
	10.4	Timers with counters	179
	10.5	Sequencer	180
		Problems	182
11 Shift registers	11.1	Shift registers	189
	11.2	Ladder programs	190
		Problems	194
12 Data handling	12.1	Registers and bits	197
	12.2	Data handling	198
	12.3	Arithmetic functions	202
	12.4	Closed loop control	203
		Problems	206
13 Designing systems	13.1	Program development	210
	13.2	Safe systems	214
	13.3	Commissioning	218
	13.4	Fault finding	220
	13.5	System documentation	227
		Problems	248
14 Programs	14.1	Temperature control	250
	14.2	Valve sequencing	254
	14.3	Conveyor belt control	265
	14.4	Control of a process	269
		Problems	271
Appendix: Symbols			276
		Answers	281
		Index	288

Preface

Technological advances in recent years have resulted in the development of the programmable logic controller and a consequential revolution of control engineering. This book is an introduction to programmable logic controllers and aims to ease the tasks of practising engineers coming first into contact with programmable logic controllers, and also provides a basic course for students on courses such as Nationals and Higher Nationals in Engineering, company training programmes and as an introduction for first year undergraduate courses in engineering.

The book has been designed to provide full syllabus coverage of the BTEC National and Higher National in Engineering units Programmable Controllers and Programmable Logic Controllers from Edexcel. It addresses the problem of different programmable control manufacturers using different nomenclature and program forms by describing the principles involved and illustrating them with examples from a range of manufacturers. The text includes:

- The basic architecture of PLCs and the characteristics of commonly used input and outputs to such systems.
- A discussion of the number systems: denary, binary, octal, hexadecimal and BCD.
- A painstaking methodical introduction, with lots of illustrations, of how to program PLCs, whatever the manufacturer, and make use of internal relays, timers, counters, shift registers, sequencers and data handling facilities.
- Consideration of the standards given by IEC 1131-3 and the programming methods of ladder, functional block diagram, instruction list, structured text and sequential function chart.
- To assist the reader to develop the skills necessary to write programs for programmable logic controllers, many worked examples, multi-choice questions and problems are included in the book with answers to all multi-choice questions and problems given at the end of the book.

Changes from third edition

The fourth edition is a complete restructuring and updating of the third edition and includes a more detailed consideration of IEC 1131-3, including all the programming methods given in the standard, and the problems of safety. This includes a discussion of emergency stop relays and safety PLCs.

Aims

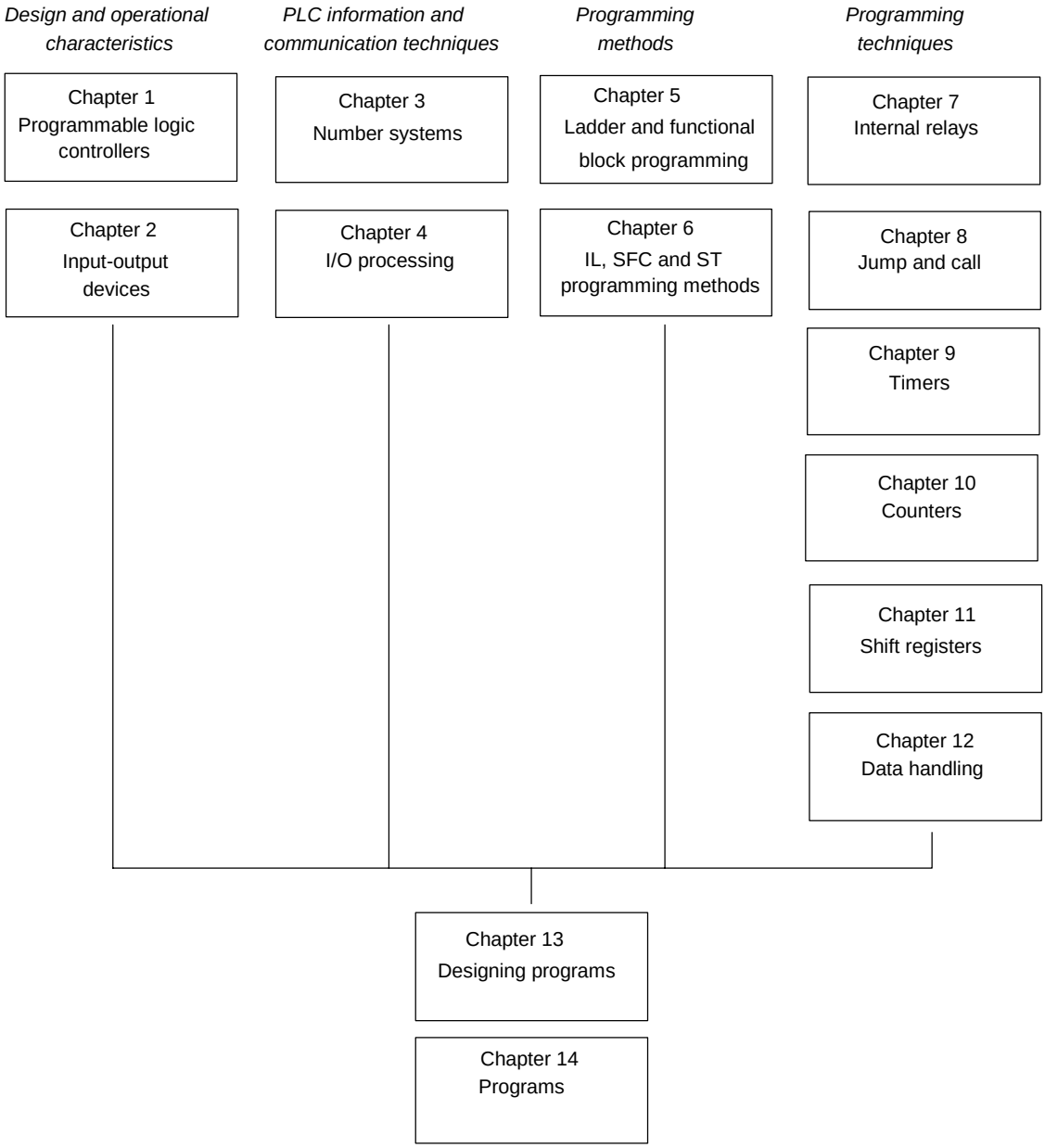
This book aims to enable the reader to:

- Identify and explain the main design characteristics, internal architecture and operating principles of programmable logic controllers.
- Describe and identify the characteristics of commonly used input and output devices.
- Explain the processing of inputs and outputs by PLCs.
- Describe communication links involved with PLC systems, the protocols and networking methods.
- Develop ladder programs for the logic functions AND, OR, NOR, NAND, NOT and XOR.
- Develop ladder programs involving internal relays, timers, counters, shift registers, sequencers and data handling.
- Develop functional block diagram, instruction list, structured text and sequential function chart programs.
- Identify safety issues with PLC systems.
- Identify methods used for fault diagnosis, testing and debugging.

Structure of the book

The figure on the following page outlines the structure of the book.

W. Bolton


Design and operational characteristics
PLC information and communication techniques
Programming methods
Programming techniques
Chapter 1 Programmable logic controllers
Chapter 3 Number systems
Chapter 5 Ladder and functional block programming
Chapter 7 Internal relays
Chapter 2 Input-output devices
Chapter 4 I/O processing
Chapter 6 IL, SFC and ST programming methods
Chapter 8 Jump and call
Chapter 9 Timers
Chapter 10 Counters
Chapter 11 Shift registers
Chapter 12 Data handling
Chapter 13 Designing programs
Chapter 14 Programs

1 Programmable logic controllers

This chapter is an introduction to the programmable logic controller, its general function, hardware forms and internal architecture. This overview is followed up by more detailed discussion in the following chapters.

1.1 Controllers

What type of task might a control system have? It might be required to control a sequence of events or maintain some variable constant or follow some prescribed change. For example, the control system for an automatic drilling machine (Figure 1.1(a)) might be required to start lowering the drill when the workpiece is in position, start drilling when the drill reaches the workpiece, stop drilling when the drill has produced the required depth of hole, retract the drill and then switch off and wait for the next workpiece to be put in position before repeating the operation. Another control system (Figure 1.1(b)) might be used to control the number of items moving along a conveyor belt and direct them into a packing case. The inputs to such control systems might be from switches being closed or opened, e.g. the presence of the workpiece might be indicated by it moving against a switch and closing it, or other sensors such as those used for temperature or flow rates. The controller might be required to run a motor to move an object to some position, or to turn a valve, or perhaps a heater, on or off.

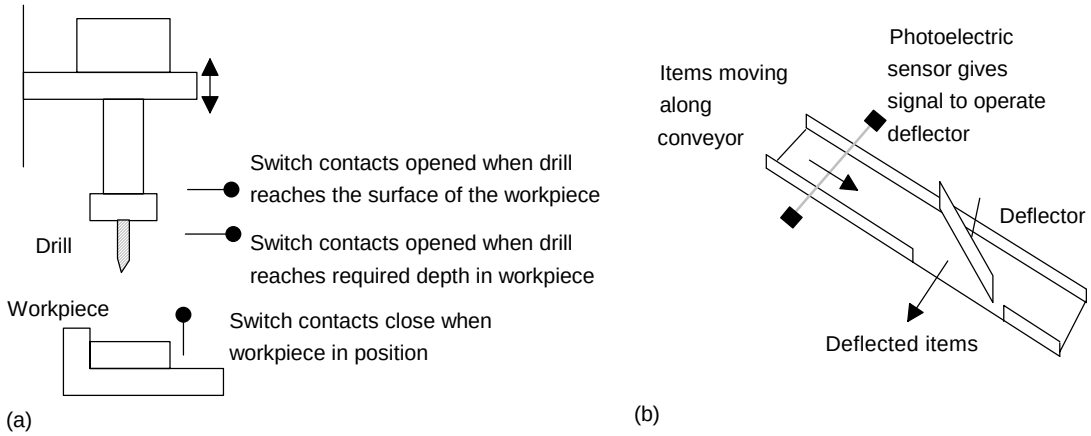

Switch contacts opened when drill reaches the surface of the workpiece
Drill
Switch contacts opened when drill reaches required depth in workpiece
Workpiece
Switch contacts close when workpiece in position
(a)
Items moving along conveyor
Photoelectric sensor gives signal to operate deflector
Deflector
Deflected items
(b)

Figure 1.1 *An example of a control task and some input sensors: (a) an automatic drilling machine, (b) a packing system*

What form might a controller have? For the automatic drilling machine, we could wire up electrical circuits in which the closing or opening of switches would result in motors being switched on or valves being actuated. Thus we might have the closing of a switch activating a relay which, in turn, switches on the current to a motor and causes the drill to rotate (Figure 1.2). Another switch might be used to activate a relay and switch on the current to a pneumatic or hydraulic valve which results in pressure being switched to drive a piston in a cylinder and so results in the workpiece being pushed into the required position. Such electrical circuits would have to be specific to the automatic drilling machine. For controlling the number of items packed into a packing case we could likewise wire up electrical circuits involving sensors and motors. However, the controller circuits we devised for these two situations would be different. In the 'traditional' form of control system, the rules governing the control system and when actions are initiated are determined by the wiring. When the rules used for the control actions are changed, the wiring has to be changed.

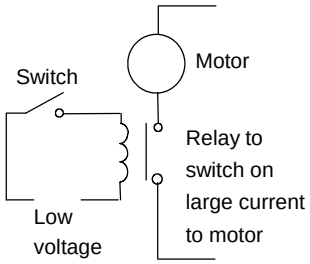


Figure 1.2 *A control circuit*

1.1.1 Microprocessor controlled system

Instead of hardwiring each control circuit for each control situation we can use the same basic system for all situations if we use a microprocessor-based system and write a program to instruct the microprocessor how to react to each input signal from, say, switches and give the required outputs to, say, motors and valves. Thus we might have a program of the form:

```
If switch A closes
Output to motor circuit
If switch B closes
Output to valve circuit
```

By changing the instructions in the program we can use the same microprocessor system to control a wide variety of situations.

As an illustration, the modern domestic washing machine uses a microprocessor system. Inputs to it arise from the dials used to select the required wash cycle, a switch to determine that the machine door is closed, a temperature sensor to determine the temperature of the water and

a switch to detect the level of the water. On the basis of these inputs the microprocessor is programmed to give outputs which switch on the drum motor and control its speed, open or close cold and hot water valves, switch on the drain pump, control the water heater and control the door lock so that the machine cannot be opened until the washing cycle is completed.

1.1.2 The programmable logic controller

A *programmable logic controller* (PLC) is a special form of microprocessor-based controller that uses a programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting and arithmetic in order to control machines and processes (Figure 1.3) and are designed to be operated by engineers with perhaps a limited knowledge of computers and computing languages. They are not designed so that only computer programmers can set up or change the programs. Thus, the designers of the PLC have pre-programmed it so that the control program can be entered using a simple, rather intuitive, form of language, see Chapter 4. The term *logic* is used because programming is primarily concerned with implementing logic and switching operations, e.g. if A or B occurs switch on C, if A and B occurs switch on D. Input devices, e.g. sensors such as switches, and output devices in the system being controlled, e.g. motors, valves, etc., are connected to the PLC. The operator then enters a sequence of instructions, i.e. a program, into the memory of the PLC. The controller then monitors the inputs and outputs according to this program and carries out the control rules for which it has been programmed.

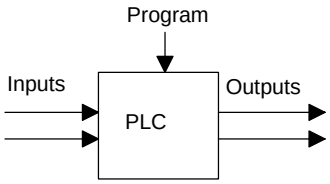


Figure 1.3 *A programmable logic controller*

PLCs have the great advantage that the same basic controller can be used with a wide range of control systems. To modify a control system and the rules that are to be used, all that is necessary is for an operator to key in a different set of instructions. There is no need to rewire. The result is a flexible, cost effective, system which can be used with control systems which vary quite widely in their nature and complexity.

PLCs are similar to computers but whereas computers are optimised for calculation and display tasks, PLCs are optimised for control tasks and the industrial environment. Thus PLCs are:

1 Rugged and designed to withstand vibrations, temperature, humidity and noise.
2 Have interfacing for inputs and outputs already inside the controller.

3 Are easily programmed and have an easily understood programming language which is primarily concerned with logic and switching operations.

The first PLC was developed in 1969. They are now widely used and extend from small self-contained units for use with perhaps 20 digital inputs/outputs to modular systems which can be used for large numbers of inputs/outputs, handle digital or analogue inputs/outputs, and also carry out proportional-integral-derivative control modes.

1.2 Hardware

Typically a PLC system has the basic functional components of processor unit, memory, power supply unit, input/output interface section, communications interface and the programming device. Figure 1.4 shows the basic arrangement.

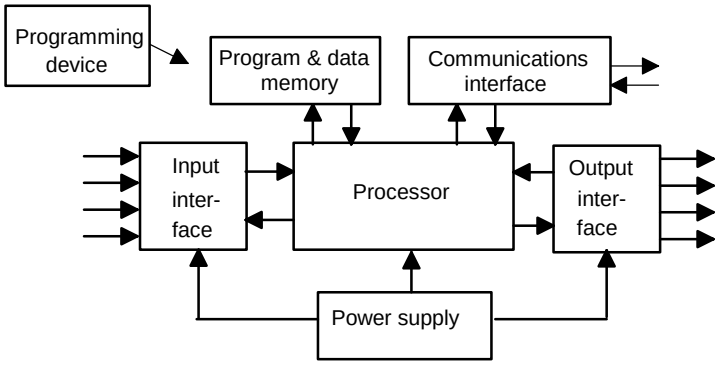


Figure 1.4 *The PLC system*

1 The *processor unit* or *central processing unit (CPU)* is the unit containing the microprocessor and this interprets the input signals and carries out the control actions, according to the program stored in its memory, communicating the decisions as action signals to the outputs.
2 The *power supply unit* is needed to convert the mains a.c. voltage to the low d.c. voltage (5 V) necessary for the processor and the circuits in the input and output interface modules.
3 The *programming device* is used to enter the required program into the memory of the processor. The program is developed in the device and then transferred to the memory unit of the PLC.
4 The *memory unit* is where the program is stored that is to be used for the control actions to be exercised by the microprocessor and data stored from the input for processing and for the output for outputting.
5 The *input and output sections* are where the processor receives information from external devices and communicates information to external devices. The inputs might thus be from switches, as illustrated in Figure 1.1(a) with the automatic drill, or other sensors such as photo-electric cells, as in the counter mechanism in Figure 1.1(b), temperature sensors, or flow sensors, etc. The outputs might be to motor starter coils, solenoid valves, etc. Input and output

interfaces are discussed in Chapter 2. Input and output devices can be classified as giving signals which are discrete, digital or analogue (Figure 1.5). Devices giving *discrete* or *digital signals* are ones where the signals are either off or on. Thus a switch is a device giving a discrete signal, either no voltage or a voltage. *Digital* devices can be considered to be essentially discrete devices which give a sequence of on–off signals. *Analogue* devices give signals whose size is proportional to the size of the variable being monitored. For example, a temperature sensor may give a voltage proportional to the temperature.

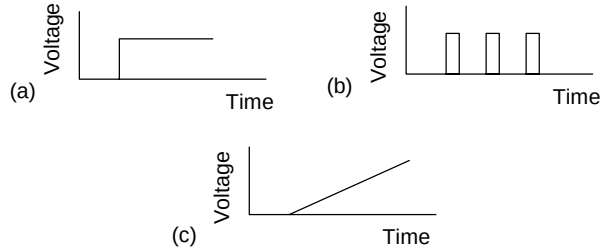


Figure 1.5 *Signals: (a) discrete, (b) digital, (c) analogue*

6 The *communications interface* is used to receive and transmit data on communication networks from or to other remote PLCs (Figure 1.6). It is concerned with such actions as device verification, data acquisition, synchronisation between user applications and connection management.

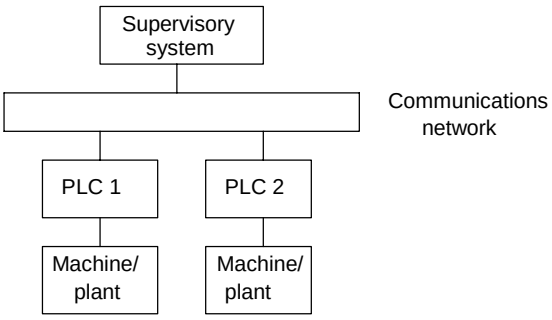


Figure 1.6 *Basic communications model*

1.3 Internal architecture

Figure 1.7 shows the basic internal architecture of a PLC. It consists of a central processing unit (CPU) containing the system microprocessor, memory, and input/output circuitry. The CPU controls and processes all the operations within the PLC. It is supplied with a clock with a frequency of typically between 1 and 8 MHz. This frequency determines the operating speed of the PLC and provides the timing and synchronisation for all elements in the system. The information within the PLC is carried by means of digital signals. The internal paths along which digital signals flow are called *buses*. In the physical sense, a bus is just a number of

conductors along which electrical signals can flow. It might be tracks on a printed circuit board or wires in a ribbon cable. The CPU uses the *data bus* for sending data between the constituent elements, the *address bus* to send the addresses of locations for accessing stored data and the *control bus* for signals relating to internal control actions. The *system bus* is used for communications between the input/output ports and the input/output unit.

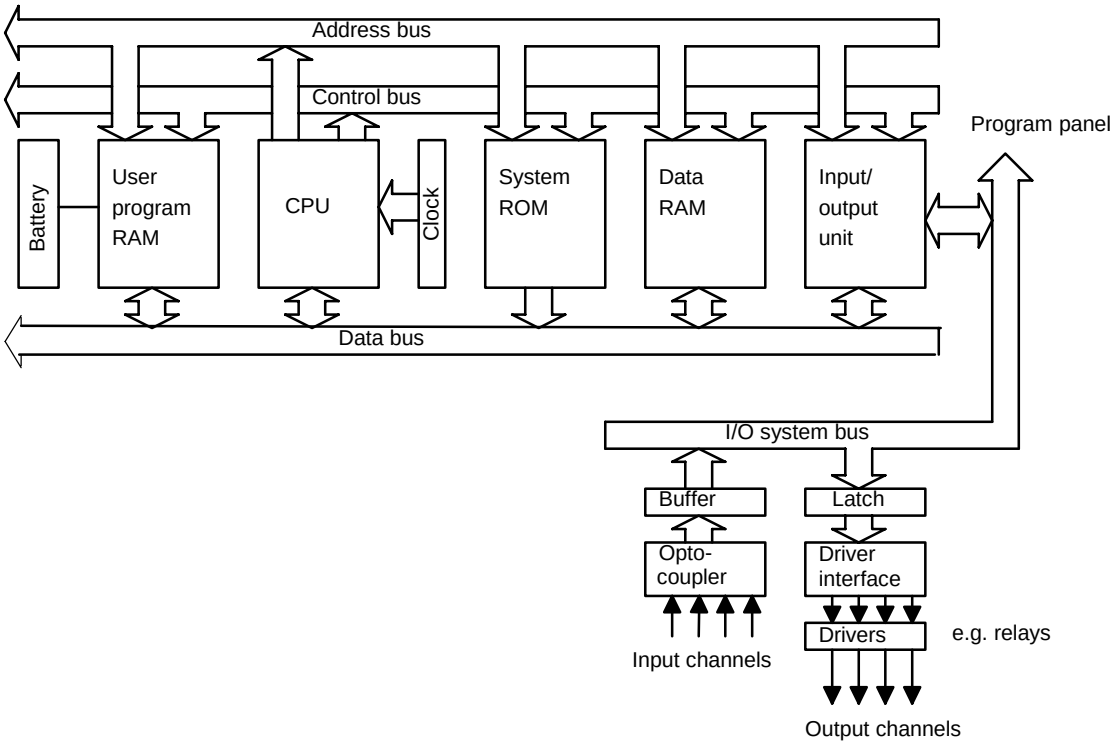


Figure 1.7 *Architecture of a PLC*

1.3.1 The CPU

The internal structure of the CPU depends on the microprocessor concerned. In general they have:

1 An *arithmetic and logic unit* (ALU) which is responsible for data manipulation and carrying out arithmetic operations of addition and subtraction and logic operations of AND, OR, NOT and EXCLUSIVE-OR.
2 Memory, termed *registers*, located within the microprocessor and used to store information involved in program execution.
3 A *control unit* which is used to control the timing of operations.

1.3.2 The buses

The buses are the paths used for communication within the PLC. The information is transmitted in binary form, i.e. as a group of *bits* with a bit

being a binary digit of 1 or 0, i.e. on/off states. The term *word* is used for the group of bits constituting some information. Thus an 8-bit word might be the binary number 00100110. Each of the bits is communicated simultaneously along its own parallel wire. The system has four buses:

1 The *data bus* carries the data used in the processing carried out by the CPU. A microprocessor termed as being 8-bit has an internal data bus which can handle 8-bit numbers. It can thus perform operations between 8-bit numbers and deliver results as 8-bit values.
2 The *address bus* is used to carry the addresses of memory locations. So that each word can be located in the memory, every memory location is given a unique *address*. Just like houses in a town are each given a distinct address so that they can be located, so each word location is given an address so that data stored at a particular location can be accessed by the CPU either to read data located there or put, i.e. write, data there. It is the address bus which carries the information indicating which address is to be accessed. If the address bus consists of 8 lines, the number of 8-bit words, and hence number of distinct addresses, is $2^8 = 256$. With 16 address lines, 65 536 addresses are possible.
3 The *control bus* carries the signals used by the CPU for control, e.g. to inform memory devices whether they are to receive data from an input or output data and to carry timing signals used to synchronise actions.
4 The *system bus* is used for communications between the input/output ports and the input/output unit.

1.3.3 Memory

There are several memory elements in a PLC system:

1 System *read-only-memory* (*ROM)* to give permanent storage for the operating system and fixed data used by the CPU.
2 *Random-access memory (RAM)* for the user's program.
3 *Random-access memory (RAM)* for data. This is where information is stored on the status of input and output devices and the values of timers and counters and other internal devices. The data RAM is sometimes referred to as a *data table* or *register table*. Part of this memory, i.e. a block of addresses, will be set aside for input and output addresses and the states of those inputs and outputs. Part will be set aside for preset data and part for storing counter values, timer values, etc.
4 Possibly, as a bolt-on extra module, *erasable and programmable read-only-memory (EPROM)* for ROMs that can be programmed and then the program made permanent.

The programs and data in RAM can be changed by the user. All PLCs will have some amount of RAM to store programs that have been developed by the user and program data. However, to prevent the loss of programs when the power supply is switched off, a battery is used in the PLC to maintain the RAM contents for a period of time. After a program

has been developed in RAM it may be loaded into an EPROM memory chip, often a bolt-on module to the PLC, and so made permanent. In addition there are temporary *buffer* stores for the input/output channels.

The storage capacity of a memory unit is determined by the number of binary words that it can store. Thus, if a memory size is 256 words then it can store $256 \times 8 = 2048$ bits if 8-bit words are used and $256 \times 16 = 4096$ bits if 16-bit words are used. Memory sizes are often specified in terms of the number of storage locations available with 1K representing the number 2^{10}, i.e. 1024. Manufacturers supply memory chips with the storage locations grouped in groups of 1, 4 and 8 bits. A 4K × 1 memory has $4 \times 1 \times 1024$ bit locations. A 4K × 8 memory has $4 \times 8 \times 1024$ bit locations. The term *byte* is used for a word of length 8 bits. Thus the 4K × 8 memory can store 4096 bytes. With a 16-bit address bus we can have 2^{16} different addresses and so, with 8-bit words stored at each address, we can have $2^{16} \times 8$ storage locations and so use a memory of size $2^{16} \times 8/2^{10} = 64\text{K} \times 8$ which we might be as four 16K × 8 bit memory chips.

1.3.4 Input/output unit

The input/output unit provides the interface between the system and the outside world, allowing for connections to be made through input/output channels to input devices such as sensors and output devices such as motors and solenoids. It is also through the input/output unit that programs are entered from a program panel. Every input/output point has a unique address which can be used by the CPU. It is like a row of houses along a road, number 10 might be the 'house' to be used for an input from a particular sensor while number '45' might be the 'house' to be used for the output to a particular motor.

The input/output channels provide isolation and signal conditioning functions so that sensors and actuators can often be directly connected to them without the need for other circuitry. Electrical isolation from the external world is usually by means of *optoisolators* (the term *optocoupler* is also often used). Figure 1.8 shows the principle of an optoisolator. When a digital pulse passes through the light-emitting diode, a pulse of infrared radiation is produced. This pulse is detected by the phototransistor and gives rise to a voltage in that circuit. The gap between the light-emitting diode and the phototransistor gives electrical isolation but the arrangement still allows for a digital pulse in one circuit to give rise to a digital pulse in another circuit.

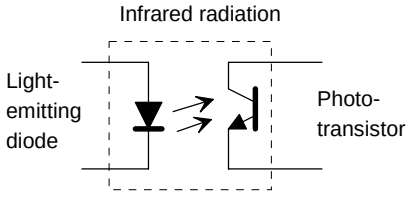


Figure 1.8 *Optoisolator*

The digital signal that is generally compatible with the microprocessor in the PLC is 5 V d.c. However, signal conditioning in the input channel,

with isolation, enables a wide range of input signals to be supplied to it (see Chapter 3 for more details). A range of inputs might be available with a larger PLC, e.g. 5 V, 24 V, 110 V and 240 V digital/discrete, i.e. on–off, signals (Figure 1.9). A small PLC is likely to have just one form of input, e.g. 24 V.

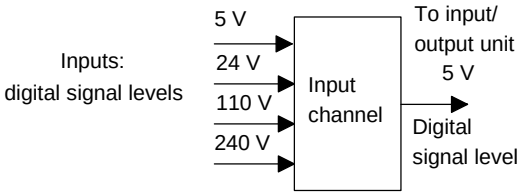


Figure 1.9 *Input levels*

The output from the input/output unit will be digital with a level of 5 V. However, after signal conditioning with relays, transistors or triacs, the output from the output channel might be a 24 V, 100 mA switching signal, a d.c. voltage of 110 V, 1 A or perhaps 240 V, 1 A a.c., or 240 V, 2 A a.c., from a triac output channel (Figure 1.10). With a small PLC, all the outputs might be of one type, e.g. 240 V a.c., 1 A. With modular PLCs, however, a range of outputs can be accommodated by selection of the modules to be used.

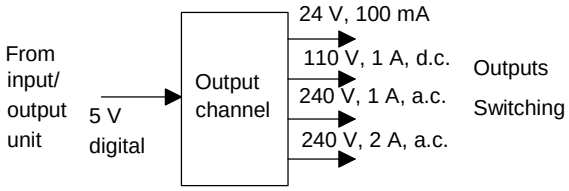


Figure 1.10 *Output levels*

Outputs are specified as being of relay type, transistor type or triac type (see Chapter 3 for more details):

1 With the *relay type*, the signal from the PLC output is used to operate a relay and is able to switch currents of the order of a few amperes in an external circuit. The relay not only allows small currents to switch much larger currents but also isolates the PLC from the external circuit. Relays are, however, relatively slow to operate. Relay outputs are suitable for a.c. and d.c. switching. They can withstand high surge currents and voltage transients.
2 The *transistor type* of output uses a transistor to switch current through the external circuit. This gives a considerably faster switching action. It is, however, strictly for d.c. switching and is destroyed by overcurrent and high reverse voltage. As a protection, either a fuse or built-in electronic protection are used. Optoisolators are used to provide isolation.

3 *Triac* outputs, with optoisolators for isolation, can be used to control external loads which are connected to the a.c. power supply. It is strictly for a.c. operation and is very easily destroyed by overcurrent. Fuses are virtually always included to protect such outputs.

1.3.5 Sourcing and sinking

The terms *sourcing* and *sinking* are used to describe the way in which d.c. devices are connected to a PLC. With sourcing, using the conventional current flow direction as from positive to negative, an input device receives current from the input module, i.e. the input module is the source of the current (Figure 1.11(a)). If the current flows from the output module to an output load then the output module is referred to as sourcing (Figure 1.11(b)). With sinking, using the conventional current flow direction as from positive to negative, an input device supplies current to the input module, i.e. the input module is the sink for the current (Figure 1.12(a)). If the current flows to the output module from an output load then the output module is referred to as sinking (Figure 1.12(b)).

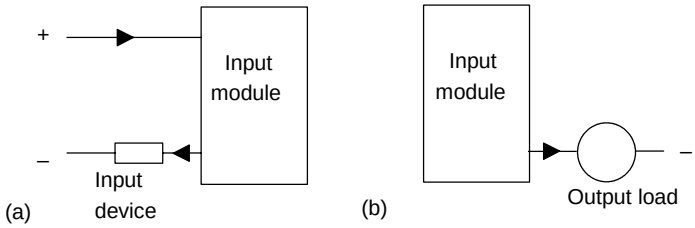


Figure 1.11 *Sourcing*

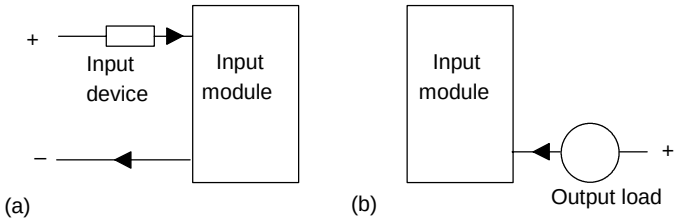


Figure 1.12 *Sinking*

1.4 PLC systems

There are two common types of mechanical design for PLC systems; a *single box*, and the *modular/rack types*. The single box type (or, as sometimes termed, brick) is commonly used for small programmable controllers and is supplied as an integral compact package complete with power supply, processor, memory, and input/output units. Typically such a PLC might have 6, 8, 12 or 24 inputs and 4, 8 or 16 outputs and a memory which can store some 300 to 1000 instructions. Figure 1.13 shows the Mitsubishi MELSEC FX3U compact, i.e. brick, PLC and Table 1.1 gives details of models in that Mitsubishi range.

Figure 1.13 *Mitsubishi Compact PLC – MELSEC FX3U (By permission of Mitsubishi Electric Europe)*

Table 1.1 *Mitsubishi Compact PLC – MELSEC FX3U Product range (By permission of Mitsubishi Electric Europe)*

Type	FX3U-16 MR	FX3U-32 MR	FX3U-48 MR	FX3U-64 MR	FX3U-80 MR
Power supply	100-240 V AC				
Inputs	8	16	24	32	40
Outputs	8	16	24	32	40
Digital outputs	Relay				
Program cycle period per logical instruction	0.065 μs				
User memory	64k steps (standard), FLROM cassettes (optional)				
Dimensions in mm (W × H × D)	130 × 90 × 86	150 × 140 × 86	182 × 90 × 86	220 × 90 × 86	285 × 90 × 86

Some brick systems have the capacity to be extended to cope with more inputs and outputs by linking input/output boxes to them. Figure 1.14 shows such an arrangement with the OMRON CPM1A PLC. The base input/output brick, depending on the model concerned, has 10, 20, 30 or 40 inputs/outputs (I/O). The 10 I/O brick has 6 d.c. input points and four outputs, the 20 I/O brick has 12 d.c. input points and 8 outputs, the 30 I/O brick has 18 d.c. input points and 12 outputs and the 40 I/O brick has 24 d.c. input points and 16 outputs. However, the 30 and 40 I/O models can be extended to a maximum of 100 inputs/outputs by linking expansion units to the original brick. For example a 20 I/O expansion module might be added, it having 12 inputs and 8 outputs, the outputs being relays, sinking transistors or sourcing transistors. Up to three expansion modules can be added. The outputs can be relay or transistor outputs.

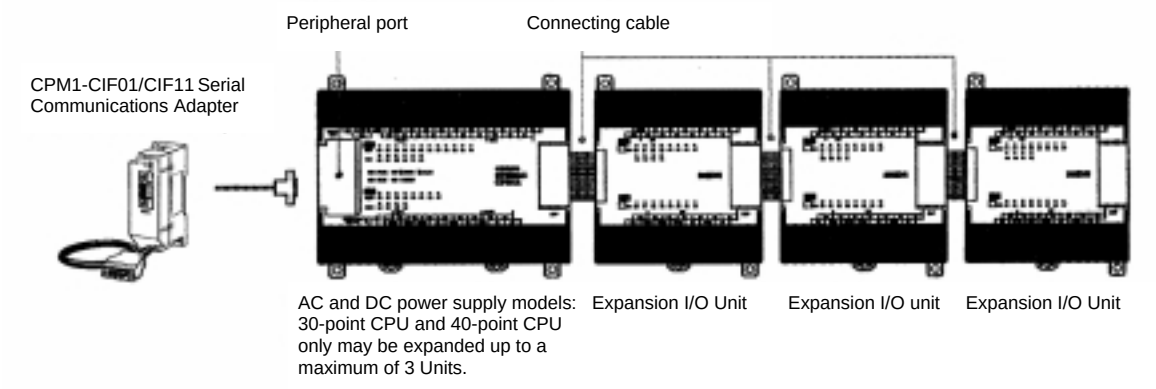

Peripheral port
Connecting cable
CPM1-CIF01/CIF11 Serial Communications Adapter
AC and DC power supply models: 30-point CPU and 40-point CPU only may be expanded up to a maximum of 3 Units.
Expansion I/O Unit
Expansion I/O unit
Expansion I/O Unit

Figure 1.14 *Basic configuration of the OMRON CPM1A PLC (By permission of Omron Electronics LLC)*

Systems with larger numbers of inputs and outputs are likely to be modular and designed to fit in racks. The modular type consists of separate modules for power supply, processor, etc., which are often mounted on rails within a metal cabinet. The rack type can be used for all sizes of programmable controllers and has the various functional units packaged in individual modules which can be plugged into sockets in a base rack. The mix of modules required for a particular purpose is decided by the user and the appropriate ones then plugged into the rack. Thus it is comparatively easy to expand the number of input/output (I/O) connections by just adding more input/output modules or to expand the memory by adding more memory units.

An example of such a modular system is provided by the Allen-Bradley PLC-5 PLC of Rockwell automation (Figure 1.15). PLC-5 processors are available in a range of I/O capacity and memory size, and can be configured for a variety of communication networks. They are single-slot modules that are placed in the left-most slot of a 1771 I/O chassis. Some 1771 I/O chassis are built for back-panel mounting and some are built for rack mounting and are available in sizes of 4, 8, 12, or 16 I/O module slots. The 1771 I/O modules are available in densities of 8, 16, or 32 I/O per module. A PLC-5 processor can communicate with I/O across a DeviceNet or Universal Remote I/O link.

A large selection of 1771 input/output modules, both digital and analogue, are available for use in the local chassis, and an even larger selection available for use at locations remote from the processor. Digital I/O modules have digital I/O circuits that interface to on/off sensors such as pushbutton and limit switches; and on/off actuators such as motor starters, pilot lights, and annunciators. Analogue I/O modules perform the required A/D and D/A conversions using up to 16-bit resolution. Analogue I/O can be user-configured for the desired fault-response state in the event that I/O communication is disrupted. This feature provides a safe reaction/response in case of a fault, limits the extent of faults, and provides a predictable fault response. 1771 I/O modules include optical coupling and filter circuitry for signal noise reduction.

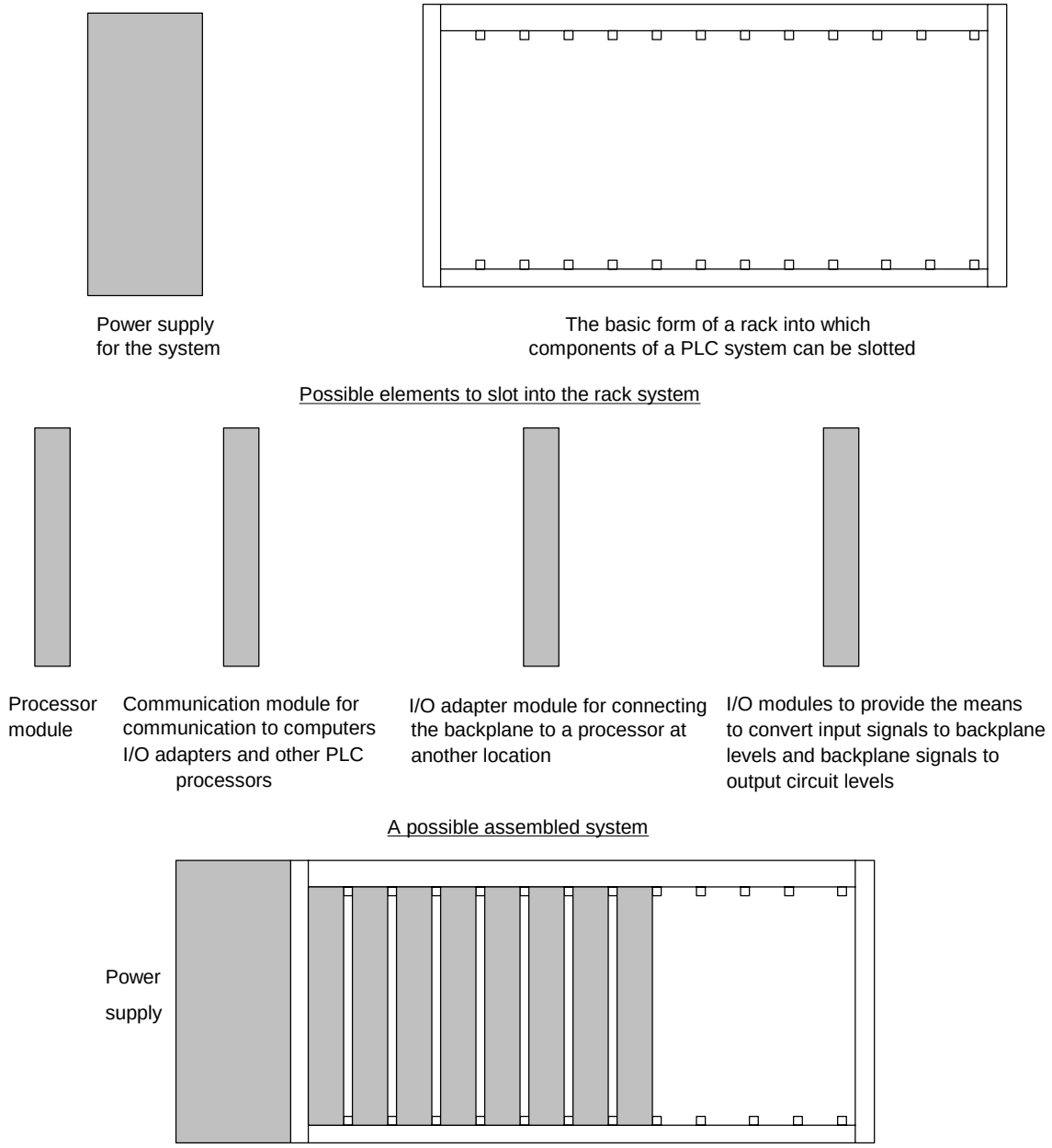


Figure 1.15 *A possible arrangement of a rack system, e.g. the Rockwell Automation , Allen-Bradley PLC-5*

Digital I/O modules cover electrical ranges from 5…276V a.c. or d.c. and relay contact output modules are available for ranges from 0…276 V ac or 0…175 V dc. A range of analogue signal levels can be accomodated, including standard analogue inputs and outputs and direct thermocouple and RTD temperature inputs.

1.4.1 Programming PLCs

Programming devices can be a hand-held device, a desktop console or a computer. Only when the program has been designed on the programming device and is ready is it transferred to the memory unit of the PLC.

1 *Hand-held programming devices* will normally contain enough memory to allow the unit to retain programs while being carried from one place to another.
2 *Desktop consoles* are likely to have a visual display unit with a full keyboard and screen display.
3 *Personal computers* are widely configured as program development work-stations. Some PLCs only require the computer to have appropriate software; others require special communication cards to interface with the PLC. A major advantage of using a computer is that the program can be stored on the hard disk or a CD and copies easily made.

PLC manufacturers have programming software for their PLCs. For example, Mitsubishi have *MELSOFT*. Their GX Developer supports all MELSEC controllers from the compact PLCs of the MELSEC FX series to the modular PLCs including MELSEC System Q and uses a Windows based environment. It supports the programming methods (see Chapter 4) of instruction list (IL), ladder diagram (LD) and sequential function chart (SFC) languages. You can switch back and forth between IL and LD at will while you are working. You can program your own function blocks and a wide range of utilities are available for configuring special function modules for the MELSEC System Q – there is no need to program special function modules, you just configure them. The package includes powerful editors and diagnostics functions for configuring MELSEC networks and hardware, and extensive testing and monitoring functions to help get applications up and running quickly and efficiently. It offers off-line simulation for all PLC types and thus enables simulation of all devices and application responses for realistic testing.

As another illustration, Siemens have *SIMATIC STEP 7*. This fully complies with the international standard IEC 61131-3 for PLC programming languages. With STEP 7, programmers can select between different programming languages. Besides ladder diagram (LAD) and function block diagram (FBD), STEP 7 Basis also includes the Instruction List (STL) programming language. Other additional options are available for IEC 61131-3 programming languages such as Structured Text (ST) called SIMATIC S7-SCL or a Sequential Function Chart (SFC) called SIMATIC S7-Graph which provides an efficient way to describe sequential control systems graphically. Features of the whole engineering system include system diagnostic capabilities, process diagnostic tools, PLC simulation, remote maintenance, and plant documentation. S7-PLCSIM is an optional package for STEP 7 that allows simulation of a SIMATIC S7 control platform and testing of a user program on a PC, enabling testing and refining prior to physical hardware installation. By testing early in a project's development, overall project quality can be improved. Installation and commissioning can thus be quicker and less

expensive as program faults can be detected and corrected early on during development.

Likewise, Rockell Automation have *RSLogix* for the Allen-Bradley PLC-5 family of PLCs, OMRON has CX-One and Telemecanique have ProWorx 32 for its Modicon range of PLCs.

Problems

Questions 1 to 6 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

1 The term PLC stands for:

A Personal logic computer.
B Programmable local computer.
C Personal logic controller.
D Programmable logic controller.

2 Decide whether each of these statements is True (T) or False (F).

A transistor output channel from a PLC:
(i) Is used for only d.c. switching.
(ii) Is isolated from the output load by an optocoupler.
Which option BEST describes the two statements?

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

3 Decide whether each of these statements is True (T) or False (F).

A relay output channel from a PLC:
(i) Is used for only d.c. switching.
(ii) Can withstand transient overloads.
Which option BEST describes the two statements?

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

4 Decide whether each of these statements is True (T) or False (F).

A triac output channel from a PLC:
(i) Is used for only a.c. output loads.
(ii) Is isolated from the output load by an optocoupler.
Which option BEST describes the two statements?

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

5 Which of the following is most likely to be the voltage level used internally in a PLC, excluding the voltage levels that might occur during conditioning in output/input channels:

A 5 V
B 24 V
C 110 V
D 240 V

6 Decide whether each of these statements is True (T) or False (F).

The reason for including optocouplers on input/output units is to:
(i) Provide a fuse mechanism which breaks the circuit if high voltages or currents occur.
(ii) Isolate the CPU from high voltages or currents.
Which option BEST describes the two statements?

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

7 Draw a block diagram showing in very general terms the main units in a PLC.

8 Draw a block diagram of a PLC showing the main functional items and how buses link them, explaining the functions of each block.

9 State the characteristics of the relay, transistor and triac types of PLC output channels.

10 How many bits can a 2K memory store?

2 Input–output devices

This chapter is a brief consideration of typical input and output devices used with PLCs. The input devices considered include digital and analogue devices such as mechanical switches for position detection, proximity switches, photoelectric switches, encoders, temperature and pressure switches, potentiometers, linear variable differential transformers, strain gauges, thermistors, thermotransistors and thermocouples. Output devices considered include relays, contactors, solenoid valves and motors.

2.1 Input devices

The term *sensor* is used for an input device that provides a usable output in response to a specified physical input. For example, a thermocouple is a sensor which converts a temperature difference into an electrical output. The term *transducer* is generally used for a device that converts a signal from one form to a different physical form. Thus sensors are often transducers, but also other devices can be transducers, e.g. a motor which converts an electrical input into rotation.

Sensors which give digital/discrete, i.e. on–off, outputs can be easily connected to the input ports of PLCs. Sensors which give analogue signals have to be converted to digital signals before inputting them to PLC ports. The following are some of the more common terms used to define the performance of sensors.

1 *Accuracy* is the extent to which the value indicated by a measurement system or element might be wrong. For example, a temperature sensor might have an accuracy of ±0.1°C. The *error* of a measurement is the difference between the result of the measurement and the true value of the quantity being measured errors can arise in a number of ways, e.g. the term *non-linearity error* is used for the error that occurs as a result of assuming a linear relationship between the input and output over the working range, i.e. a graph of output plotted against input is assumed to give a straight line. Few systems or elements, however, have a truly linear relationship and thus errors occur as a result of the assumption of linearity (Figure 2.1(a)). The term *hysteresis error* (Figure 2.1(b)) is used for the difference in outputs given from the same value of quantity being measured according to whether that value has been reached by a continuously increasing change or a continuously decreasing change. Thus, you might obtain a different value from a thermometer used to measure the same temperature of a liquid if it is reached by the liquid warming up to the measured temperature or it is reached by the liquid cooling down to the measured temperature.

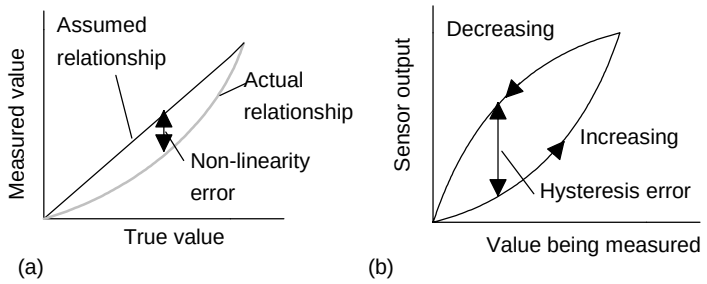


Figure 2.1 *Some sources of error: (a) non-linearity, (b) hysteresis*

2 The *range* of variable of system is the limits between which the input can vary. For example, a resistance temperature sensor might be quoted as having a range of −200 to +800ºC.

3 When the input value to a sensor changes, it will take some time to reach and settle down to the steady-state value (Figure 2.2). The *response time* is the time which elapses after the input to a system or element is abruptly increased from zero to a constant value up to the point at which the system or element gives an output corresponding to some specified percentage, e.g. 95%, of the value of the input. The *rise time* is the time taken for the output to rise to some specified percentage of the steady-state output. Often the rise time refers to the time taken for the output to rise from 10% of the steady-state value to 90 or 95% of the steady-state value. The *settling time* is the time taken for the output to settle to within some percentage, e.g. 2%, of the steady-state value.

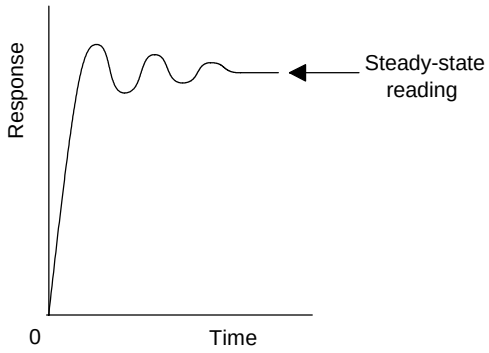


Figure 2.2 *Response of a sensor or measurement system to a sudden input. You can easily see such a response when the current in an electrical circuit is suddenly switched on and an ammeter reading observed.*

4 The *sensitivity* indicates how much the output of an instrument system or system element changes when the quantity being measured changes by a given amount, i.e. the ratio ouput/input. For example, a thermocouple might have a sensitivity of 20 μV/ºC and so give an output of 20 μV for each 1ºC change in temperature.

5 The *stability* of a system is its ability to give the same output when used to measure a constant input over a period of time. The term *drift* is often used to describe the change in output that occurs over time. The drift may be expressed as a percentage of the full range output. The term *zero drift* is used for the changes that occur in output when there is zero input.

6 The term *repeatability* is used for the ability of a measurement system to give the same value for repeated measurements of the same value of a variable. Common cause of lack of repeatability are random fluctuations in the environment, e.g. changes in temperature and humidity. The error arising from repeatability is usually expressed as a percentage of the full range output. For example, a pressure sensor might be quoted as having a repeatability of ±0.1% of full range. Thus with a range of 20 kPa this would be an error of ±20 Pa.

7 The *reliability* of a measurement system, or element in such a system, is defined as being the probability that it will operate to an agreed level of performance, for a specified period, subject to specified environmental conditions. The agreed level of performance might be that the measurement system gives a particular accuracy.

The following are examples of some of the commonly used PLC input devices and their sensors.

2.1.1 Mechanical switches

A mechanical switch generates an on–off signal or signals as a result of some mechanical input causing the switch to open or close. Such a switch might be used to indicate the presence of a workpiece on a machining table, the workpiece pressing against the switch and so closing it. The absence of the workpiece is indicated by the switch being open and its presence by it being closed. Thus, with the arrangement shown in Figure 2.3(a), the input signals to a single input channel of the PLC are thus the logic levels:

Workpiece not present 0
Workpiece present 1

The 1 level might correspond to a 24 V d.c. input, the 0 to a 0 V input.

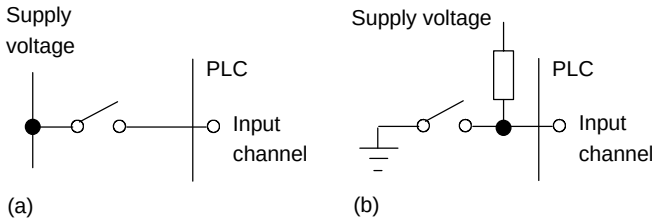


Figure 2.3 *Switch sensors*

With the arrangement shown in Figure 2.3(b), when the switch is open the supply voltage is applied to the PLC input, when the switch is closed the input voltage drops to a low value. The logic levels are thus:

Workpiece not present 1
Workpiece present 0

Switches are available with *normally open (NO)* or *normally closed (NC)* contacts or can be configured as either by choice of the relevant contacts. An NO switch has its contacts open in the absence of a mechanical input and the mechanical input is used to close the switch. An NC switch has its contacts closed in the absence of a mechanical input and the mechanical input is used to open the switch.

The term *limit switch* is used for a switch which is used to detect the presence or passage of a moving part. It can be actuated by a cam, roller or lever. Figure 2.4 shows some examples. The cam (Figure 2.4(c)) can be rotated at a constant rate and so switch the switch on and off for particular time intervals.

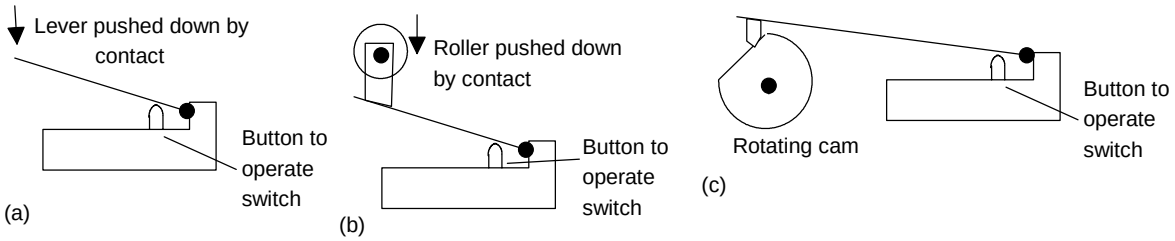


Figure 2.4 *Limit switches actuated by: (a) lever, (b) roller, (c) cam*

2.1.2 Proximity switches

Proximity switches are used to detect the presence of an item without making contact with it. There are a number of forms of such switches, some being only suitable for metallic objects.

The *eddy current* type of proximity switch has a coil which is energised by a constant alternating current and produces a constant alternating magnetic field. When a metallic object is close to it, eddy currents are induced in it (Figure 2.5(a)). The magnetic field due to these eddy currents induces an e.m.f. back in the coil with the result that the voltage amplitude needed to maintain the constant coil current changes. The voltage amplitude is thus a measure of the proximity of metallic objects. The voltage can be used to activate an electronic switch circuit, basically a transistor which has its output switched from low to high by the voltage change, and so give an on–off device. The range over which such objects can be detected is typically about 0.5 to 20 mm.

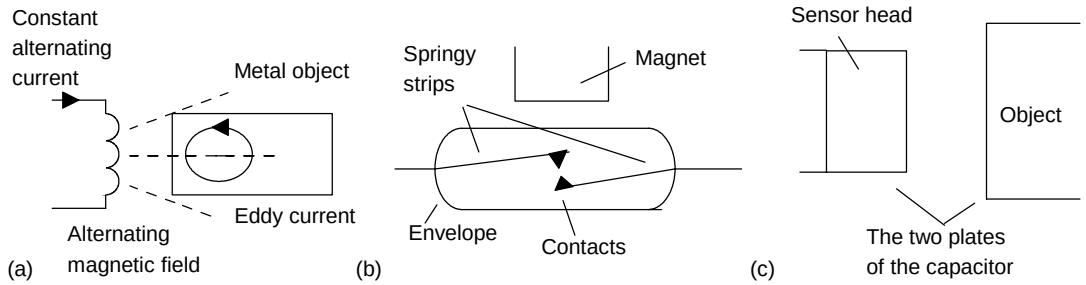


Figure 2.5 *Proximity switches: (a) eddy current, (b) reed switch*, (c) *capacitive*

Another type is the *reed switch*. This consists of two overlapping, but not touching, strips of a springy ferromagnetic material sealed in a glass or plastic envelope (Figure 2.5(b)). When a magnet or current-carrying coil is brought close to the switch, the strips become magnetised and attract each other. The contacts then close. The magnet closes the contacts when it is typically about 1 mm from the switch. Such a switch is widely used with burglar alarms to detect when a door is opened; the magnet being in the door and the reed switch in the frame of the door. When the door opens the switch opens.

A proximity switch that can be used with metallic and non-metallic objects is the *capacitive proximity switch*. The capacitance of a pair of plates separated by some distance depends on the separation, the smaller the separation the higher the capacitance. The sensor of the capacitive proximity switch is just one of the plates of the capacitor, the other plate being the metal object whose proximity is to be detected (Figure 2.5(c)). Thus the proximity of the object is detected by a change in capacitance. The sensor can also be used to detect non-metallic objects since the capacitance of a capacitor depends on the dielectric between its plates. In this case the plates are the sensor and the earth and the non-metallic object is the dielectric. The change in capacitance can be used to activate an electronic switch circuit and so give an on–off device. Capacitive proximity switches can be used to detect objects when they are typically between 4 and 60 mm from the sensor head.

Another type, the *inductive proximity switch*, consists of a coil wound round a ferrous metallic core. When one end of this core is placed near to a ferrous metal object there is effectively a change in the amount of metallic core associated with the coil and so a change in its inductance. This change in inductance can be monitored using a resonant circuit, the presence of the ferrous metal object thus changing the current in that circuit. The current can be used to activate an electronic switch circuit and so give an on–off device. The range over which such objects can be detected is typically about 2 to 15 mm.

2.1.3 Photoelectric sensors and switches

Photoelectric switch devices can either operate as *transmissive types* where the object being detected breaks a beam of light, usually infrared radiation, and stops it reaching the detector (Figure 2.6(a)) or *reflective types* where the object being detected reflects a beam of light onto the detector (Figure 2.6(b)). In both types the radiation emitter is generally a *light-emitting diode (LED)*. The radiation detector might be a *photo-transistor*, often a pair of transistors, known as a *Darlington pair*. The Darlington pair increases the sensitivity. Depending on the circuit used, the output can be made to switch to either high or low when light strikes the transistor. Such sensors are supplied as packages for sensing the presence of objects at close range, typically at less than about 5 mm. Figure 2.6(c) shows a U-shaped form where the object breaks the light beam.

Another possibility is a *photodiod*e. Depending on the circuit used, the output can be made to switch to either high or low when light strikes the diode. Yet another possibility is a *photoconductive cell*. The resistance of

the photoconductive cell, often cadmium sulphide, depends on the intensity of the light falling on it.

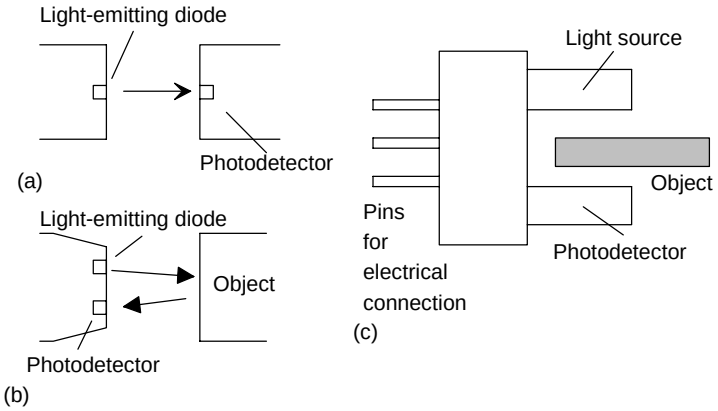


Figure 2.6 *Photoelectric sensors*

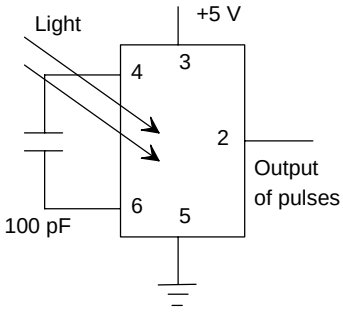


Figure 2.7 *TSL220*

With the above sensors, light is converted to a current, voltage or resistance change. If the output is to be used as a measure of the intensity of the light, rather than just the presence or absence of some object in the light path, the signal will need amplification and then conversion from analogue to digital by an analogue-to-digital converter. An alternative to this is to use a light-to-frequency converter, the light then being converted to a sequence of pulses with the frequency of the pulses being a measure of the light intensity. Integrated circuit sensors are available, e.g. the Texas Instrument TSL220, incorporating the light sensor and the voltage-to-frequency converter (Figure 2.7).

2.1.4 Encoders

The term *encoder* is used for a device that provides a digital output as a result of angular or linear displacement. An *increment encoder* detects changes in angular or linear displacement from some datum position, while an *absolute encoder* gives the actual angular or linear position.

Figure 2.8 shows the basic form of an *incremental encoder* for the measurement of angular displacement. A beam of light, from perhaps a light-emitting diode (LED), passes through slots in a disc and is detected by a light sensor, e.g. a photodiode or phototransistor. When the disc rotates, the light beam is alternately transmitted and stopped and so a pulsed output is produced from the light sensor. The number of pulses is proportional to the angle through which the disc has rotated, the resolution being proportional to the number of slots on a disc. With 60 slots then, since one revolution is a rotation of 360°, a movement from one slot to the next is a rotation of 6°. By using offset slots it is possible to have over a thousand slots for one revolution and so much higher resolution.

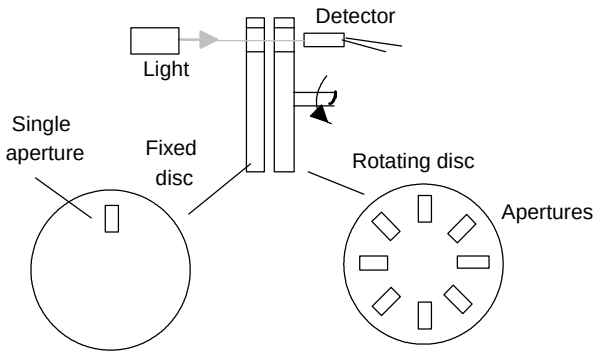


Figure 2.8 *Basic form of an incremental encoder*

The *absolute encoder* differs from the incremental encoder in having a pattern of slots which uniquely defines each angular position. With the form shown in Figure 2.9, the rotating disc has four concentric circles of slots and four sensors to detect the light pulses. The slots are arranged in such a way that the sequential output from the sensors is a number in the binary code, each such number corresponding to a particular angular position. With 4 tracks there will be 4 bits and so the number of positions that can be detected is $2^4 = 16$, i.e. a resolution of $360/16 = 22.5^o$. Typical encoders tend to have up to 10 or 12 tracks. The number of bits in the binary number will be equal to the number of tracks. Thus with 10 tracks there will be 10 bits and so the number of positions that can be detected is 2^{10}, i.e. 1024, a resolution of $360/1024 = 0.35^o$.

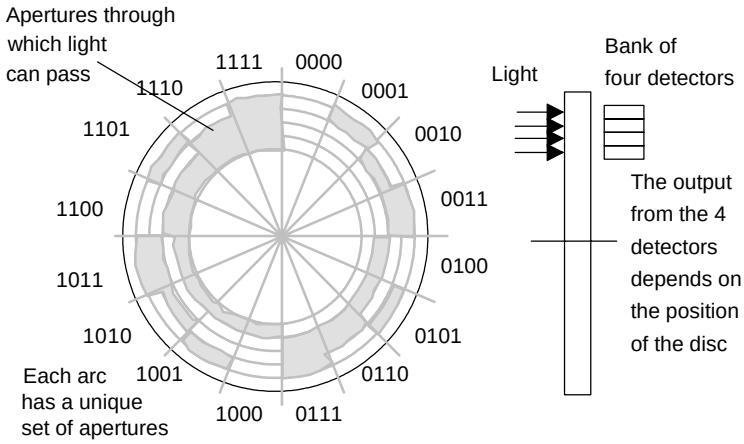


Figure 2.9 *The rotating wheel of the absolute encoder. Note that though the normal form of binary code is shown in the figure, in practice a modified form of binary code called the Gray code is generally used. This code, unlike normal binary, has only one bit changing in moving from one number to the next. Thus we have the sequence 0000, 0001, 0011, 0010, 0011, 0111, 0101, 0100, 1100, 1101, 1111.*

2.1.5 Temperature sensors

A simple form of temperature sensor which can be used to provide an on–off signal when a particular temperature is reached is the *bimetal element*. This consists of two strips of different metals, e.g. brass and iron, bonded together (Figure 2.10). The two metals have different coefficients of expansion. Thus when the temperature of the bimetal strip increases the strip curves, in order that one of the metals can expand more than the other. The higher expansion metal is on the outside of the curve. As the strip cools, the bending effect is reversed. This movement of the strip can be used to make or break electrical contacts and hence, at some particular temperature, give an on–off current in an electrical circuit. The device is not very accurate but is commonly used in domestic central heating thermostats.

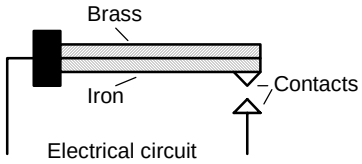


Figure 2.10 *Bimetallic strip*

Another form of temperature sensor is the *resistive temperature detector (RTD)*. The electrical resistance of metals or semiconductors changes with temperature. In the case of a metal, the ones most commonly used are platinum, nickel or nickel alloys, the resistance of which varies in a linear manner with temperature over a wide range of temperatures, though the actual change in resistance per degree is fairly small. Semiconductors, such as thermistors, show very large changes in resistance with temperature. The change, however, is non-linear. Such detectors can be used as one arm of a Wheatstone bridge and the output of the bridge taken as a measure of the temperature (Figure 2.11(a)). Another possibility is to use a potential divider circuit with the change in resistance of the thermistor changing the voltage drop across a resistor (Figure 2.11(b)). The output from either type of circuit is an analogue signal which is a measure of the temperature.

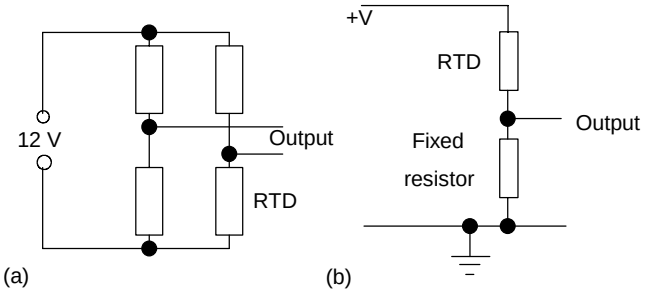


Figure 2.11 *(a) Wheatstone bridge, (b) potential divider circuits*

Thermodiodes and *thermotransistors* are used as temperature sensors since the rate at which electrons and holes diffuse across semiconductor

junctions is affected by the temperature. Integrated circuits are available which combine such a temperature-sensitive element with the relevant circuitry to give an output voltage related to temperature. A widely used integrated package is the LM35 which gives an output of 10 mV/ºC when the supply voltage is +5 V (Figure 2.12(a)).

A digital temperature switch can be produced with an analogue sensor by feeding the analogue output into a comparator amplifier which compares it with some set value, producing an output giving a logic 1 signal when the temperature voltage input is equal to or greater than the set point and otherwise an output which gives a logic 0 signal. Integrated circuits, e.g. LM3911N, are available, combining a thermotransistor temperature-sensitive element with an operational amplifier. When the connections to the chip are so made that the amplifier is connected as a comparator (Figure 2.12(b)), then the output will switch as the temperature traverses the set point and so directly give an on–off temperature controller.

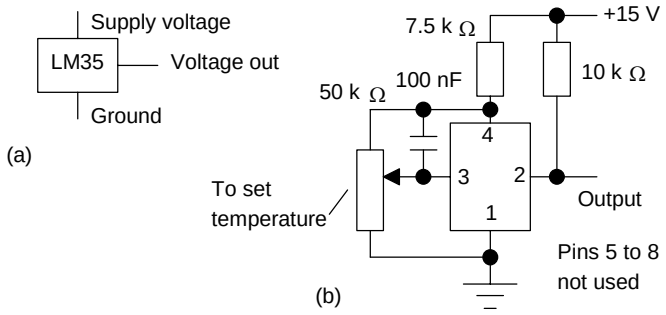


Figure 2.12 *(a) LM35, (b) LM3911N circuit for on–off control*

Another commonly used temperature sensor is the *thermocouple*. The thermocouple consists essentially of two dissimilar wires A and B forming a junction (Figure 2.13). When the junction is heated so that it is at a higher temperature than the other junctions in the circuit, which remain at a constant cold temperature, an e.m.f. is produced which is related to the hot junction temperature. The voltage produced by a thermocouple is small and needs amplification before it can be fed to the analogue channel input of a PLC. There is also circuitry required to compensate for the temperature of the cold junction since its temperature affects the value of the e.m.f. given by the hot junction. The amplification and compensation, together with filters to reduce the effect of interference from the 50 Hz mains supply, are often combined in a signal processing unit.

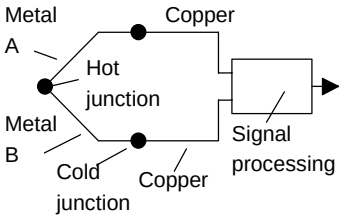


Figure 2.13 *Thermocouple*

2.1.6 Position/displacement sensors

The term *position sensor* is used for a sensor that gives a measure of the distance between a reference point and the current location of the target, a *displacement sensor* being one that gives a measure of the distance between the present position of the target and the previously recorded position.

Resistive linear and angular position sensors are widely used and relatively inexpensive. These are also called *linear and rotary potentiometers*. A d.c. voltage is provided across the full length of the track and the voltage signal between a contact which slides over the resistance track and one end of the track is related to the position of the sliding contact between the ends of the potentiometer resistance track (Figure 2.14). The potentiometer thus provides an analogue linear or angular position sensor.

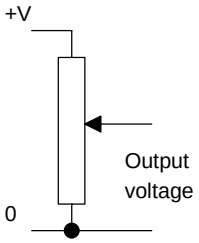


Figure 2.14 *Potentiometer*

Another form of displacement sensor is the *linear variable differential transformer (LVDT)*, this giving a voltage output related to the position of a ferrous rod. The LVDT consists of three symmetrically placed coils through which the ferrous rod moves (Figure 2.15).

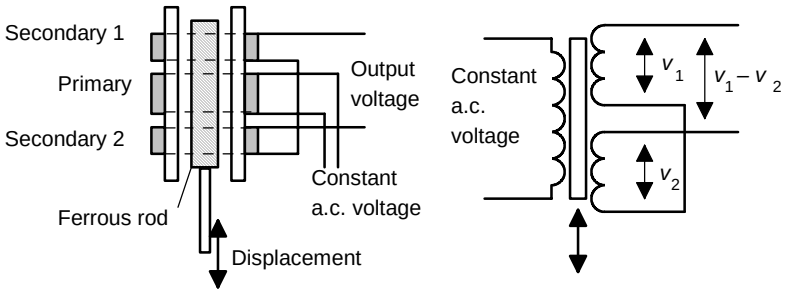


Figure 2.15 *LVDT*

When an alternating current is applied to the primary coil, alternating voltages, v_1 and v_2, are induced in the two secondary coils. When the ferrous rod core is centred between the two secondary coils, the voltages induced in them are equal. The outputs from the two secondary coils are connected so that their combined output is the difference between the two voltages, i.e. $v_1 - v_2$. With the rod central, the two alternating voltages are equal and so there is no output voltage. When the rod is displaced from its central position there is more of the rod in one secondary coil than the other. As a result the size of the alternating voltage induced in one coil is greater than that in the other. The difference between the two secondary coil voltages, i.e. the output, thus depends on the position of the ferrous rod. The output from the LVDT is an alternating voltage. This is usually converted to an analogue d.c. voltage and amplified before inputting to the analogue channel of a PLC.

Capacitive displacement sensors are essentially just parallel plate capacitors. The capacitance will change if the plate separation changes, the area of overlap of the plates changes, or a slab of dielectric is moved into or out of the plates (Figure 2.16). All these methods can be used to give linear displacement sensors. The change in capacitance has to be converted into a suitable electrical signal by signal conditioning.

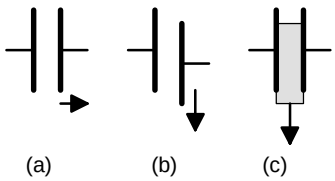


Figure 2.16 *Capacitor sensors: (a) changing the plate separation, (b) changing the area of overlap, (c) moving the dielectric*

2.1.7 Strain gauges

When a wire or strip of semiconductor is stretched, its resistance changes. The fractional change in resistance is proportional to the fractional change in length, i.e. strain.

$$\frac{\Delta R}{R} = G \times \text{strain}$$

where ΔR is the change in resistance for a wire of resistance R and G is a constant called the *gauge factor*. For metals the gauge factor is about 2 and for semiconductors about 100. Metal resistance strain gauges are in the form of a flat coil in order to get a reasonable length of metal in a small area. Often they are etched from metal foil (Figure 2.17(a)) and attached to a backing of thin plastic film so that they can be stuck on surfaces, like postage stamps on an envelope. The change in resistance of the strain gauge, when subject to strain, is usually converted into a voltage signal by the use of a Wheatstone bridge (Figure 2.17(b)). A problem that occurs is that the resistance of the strain gauge also changes with temperature and thus some means of temperature compensation has to be used so that the output of the bridge is only a function of the strain. This can be achieved by placing a dummy strain gauge in an opposite arm of the bridge, that gauge not being subject to any strain but only the temperature (Figure 2.18).

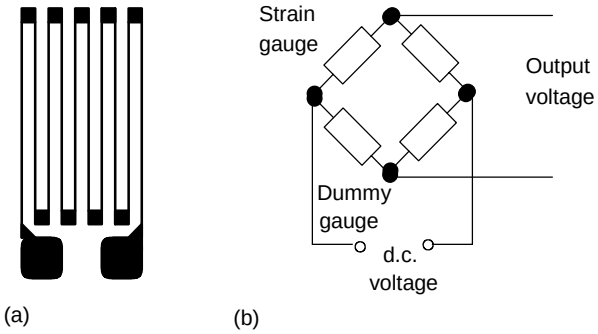


Figure 2.17 *(a) Metal foil strain gauge, (b) Wheatstone bridge circuit with compensation for temperature changes*

An alternative which is widely used is to use four active gauges as the arms of the bridge and arrange it so that one pair of opposite gauges are in tension and the other pair in compression. This not only gives temperature compensation but also gives a much larger output change when strain is applied. The following paragraph illustrates systems employing such a form of compensation.

By attaching strain gauges to other devices, changes which result in strain of those devices can be transformed, by the strain gauges, to give voltage changes. They might, for example, be attached to a cantilever to which forces are applied at its free end (Figure 2.18(a)). The voltage change, resulting from the strain gauges and the Wheatstone bridge, then becomes a measure of the force. Another possibility is to attach strain

gauges to a diaphragm which deforms as a result of pressure (Figure 2.18(b)). The output from the gauges, and associated Wheatstone bridge, then becomes a measure of the pressure.

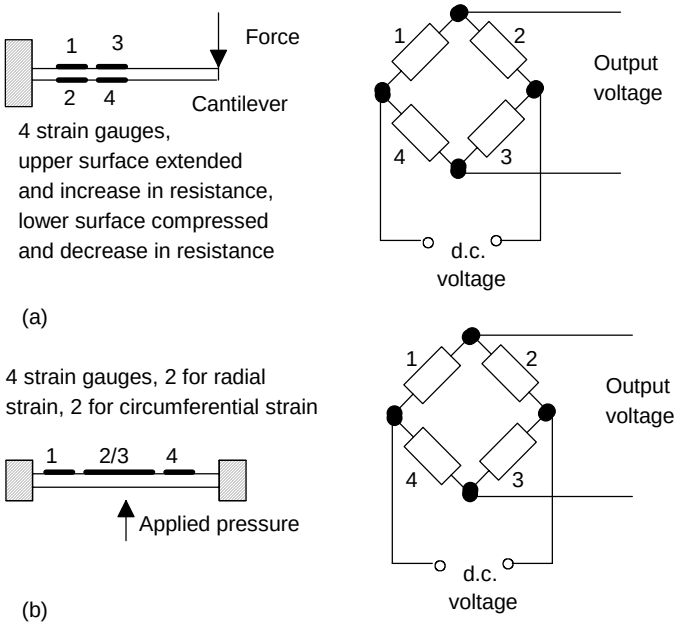


Figure 2.18 *Strain gauges used for (a) force sensor, (b) pressure sensor*

2.1.8 Pressure sensors

Commonly used pressure sensors which give responses related to the pressure are diaphragm and bellows types. The diaphragm type consists of a thin disc of metal or plastic, secured round its edges. When there is a pressure difference between the two sides of the diaphragm, the centre of it deflects. The amount of deflection is related to the pressure difference. This deflection may be detected by strain gauges attached to the diaphragm (see Figure 2.18(b)), by a change in capacitance between it and a parallel fixed plate or by using the deflection to squeeze a piezoelectric crystal (Figure 2.19(a)). When a piezoelectric crystal is squeezed, there is a relative displacement of positive and negative charges within the crystal and the outer surfaces of the crystal become charged. Hence a potential difference appears across it. An example of such a sensor is the Motorola MPX100AP sensor (Figure 2.19(b)). This has a built-in vacuum on one side of the diaphragm and so the deflection of the diaphragm gives a measure of the absolute pressure applied to the other side of the diaphragm. The output is a voltage which is proportional to the applied pressure with a sensitivity of 0.6 mV/kPa. Other versions are available which have one side of the diaphragm open to the atmosphere and so can be used to measure gauge pressure; others allow pressures to be applied to both sides of the diaphragm and so can be used to measure differential pressures.

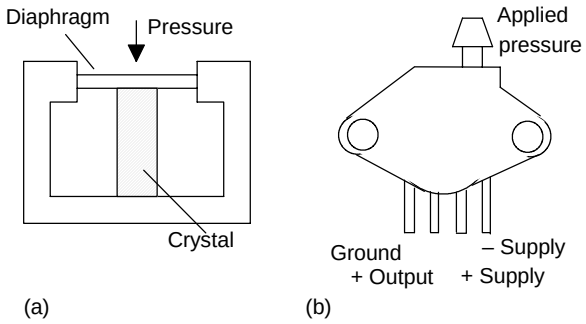


Figure 2.19 *(a) Piezoelectric pressure sensor, (b) MPX100AP*

Pressure switches are designed to switch on or off at a particular pressure. A typical form involves a diaphragm or bellows which moves under the action of the pressure and operates a mechanical switch. Figure 2.20 shows two possible forms. Diaphragms are less sensitive than bellows but can withstand greater pressures.

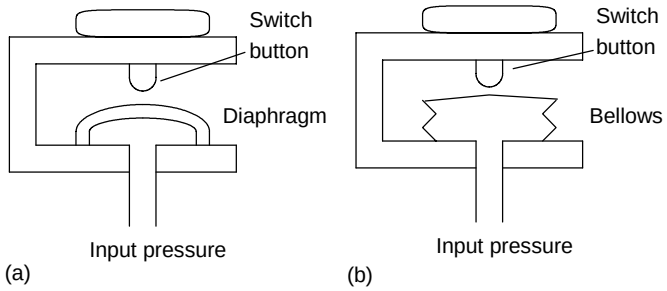


Figure 2.20 *Examples of pressure switches*

2.1.9 Liquid level detector

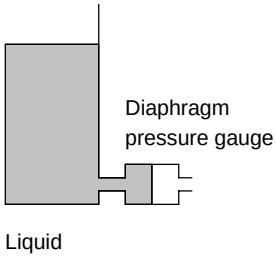


Figure 2.21 *Liquid level sensor*

Pressure sensors may be used to monitor the depth of a liquid in a tank. The pressure due to a height of liquid h above some level is $h\rho g$, where ρ is the density of the liquid and g the acceleration due to gravity. Thus a commonly used method of determining the level of liquid in a tank is to measure the pressure due to the liquid above some datum level (Figure 2.21).

Often a sensor is just required to give a signal when the level in some container reaches a particular level. A float switch that is used for this purpose consists of a float containing a magnet which moves in a housing with a reed switch. As the float rises or falls it turns the reed switch on or off, the reed switch being connected in a circuit which then switches on or off a voltage.

2.1.10 Fluid flow measurement

A common form of fluid flow meter is that based on measuring the difference in pressure resulting when a fluid flows through a constriction. Figure 2.22 shows a commonly used form, the *orifice flow meter*. As a result of the fluid flowing through the orifice, the pressure at A is higher

than that at B, the difference in pressure being a measure of the rate of flow. This pressure difference can be monitored by means of a diaphragm pressure gauge and thus becomes a measure of the rate of flow.

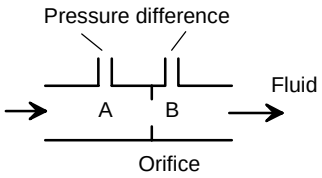


Figure 2.22 *Orifice flow meter*

2.1.11 Smart sensors

The term *smart sensor* is used for a sensor which is integrated with the required buffering and conditioning circuitry in a single element. The circuitry with the element usually consists of data converters, a processor and firmware, and some form of non-volatile EEPROM memory (electrically erasable programmable read only memory, it is similar to EPROM – see Chapter 1). The term non-volatile is used because the memory has to retain certain parameters when the power supply is removed.

Because the elements are processor-based devices, such sensors can be programmed for specific requirements. For example, it can be programmed to process the raw input data, correcting for such things as non-linearities, and then send the processed data to a base station. It can be programmed to send a warning signal when the measured parameter reaches some critical value.

The IEEE 1451.4 standard interface for smart sensors and actuators is based on an electronic data sheet (TEDS) format which is aimed at allowing installed analogue transducers to be easily connected to digital measurement systems. The standard requires the non-volatile EEPROM embedded memory to hold and communicate data which will allow a plug-and-play capability. It thus would hold data for the identification and properties for the sensor and might also contain the calibration template, so facilitating digital interrogation.

2.2 Output devices

The output ports of a PLC are of the relay type or optoisolator with transistor or triac types depending on the devices connected to them which are to be switched on or off. Generally, the digital signal from an output channel of a PLC is used to control an actuator which in turn controls some process. The term *actuator* is used for the device which transforms the electrical signal into some more powerful action which then results in the control of the process. The following are some examples.

2.2.1 Relay

Solenoids form the basis of a number of output control actuators. When a current passes through a solenoid a magnetic field is produced and this can then attract ferrous metal components in its vicinity. One example of such an actuator is the *relay*, the term *contactor* being used when large currents are involved. When the output from the PLC is switched on, the

solenoid magnetic field is produced and pulls on the contacts and so closes a switch or switches (Figure 2.23). The result is that much larger currents can be switched on. Thus the relay might be used to switch on the current to a motor.

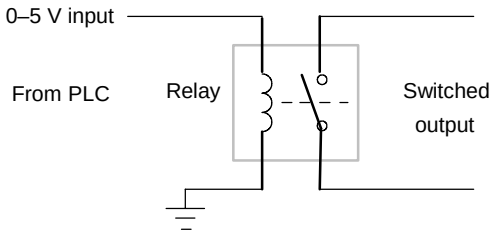


Figure 2.23 *Relay used as an output device*

2.2.2 Directional control valves

Another example of the use of a solenoid as an actuator is a *solenoid operated valve*. The valve may be used to control the directions of flow of pressurised air or oil and so used to operate other devices such as a piston moving in a cylinder. Figure 2.24 shows one such form, a *spool valve*, used to control the movement of a piston in a cylinder. Pressurised air or hydraulic fluid is inputted from port P, this being connected to the pressure supply from a pump or compressor and port T is connected to allow hydraulic fluid to return to the supply tank or, in the case of a pneumatic system, to vent the air to the atmosphere. With no current through the solenoid (Figure 2.24(a)) the hydraulic fluid of pressurised air is fed to the right of the piston and exhausted from the left, the result then being the movement of the piston to the left. When a current is passed through the solenoid, the spool valve switches the hydraulic fluid or pressurised air to the left of the piston and exhausted from the right. The piston then moves to the right. The movement of the piston might be used to push a deflector to deflect items off a conveyor belt (see Figure 1.1(b)) or implement some other form of displacement which requires power.

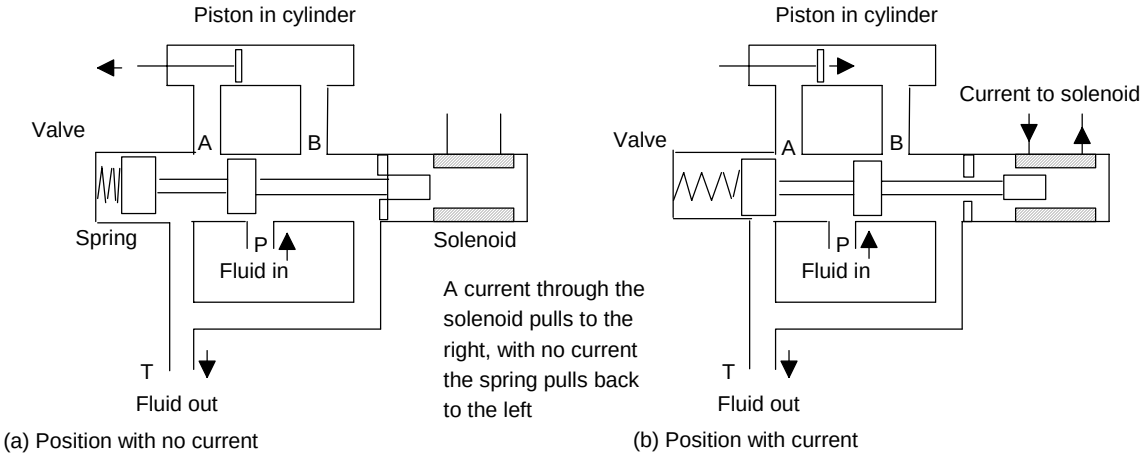


Figure 2.24 *An example of a solenoid operated valve*

With the above valve there are the two control positions shown in Figure 2.24(a) and (b). Directional control valves are described by the number of ports they have and the number of control positions. The valve shown in Figure 2.24 has four ports, i.e. A, B, P and T, and two control positions. It is thus referred to as a 4/2 valve. The basic symbol used on drawings for valves is a square, with one square being used to describe each of the control positions. Thus the symbol for the valve in Figure 2.24 consists of two squares (Figure 2.25(a)). Within each square the switching positions are then described by arrows to indicate a flow direction or a terminated line to indicate no flow path. Figure 2.25(b) shows this for the valve shown in Figure 2.24. Figure 2.26 shows some more examples of direction valves and their switching positions.

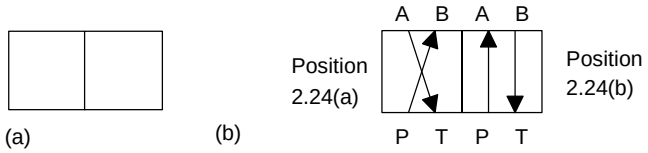


Figure 2.25 *(a) The basic symbol for a two position valve, (b) the 4/2 valve*

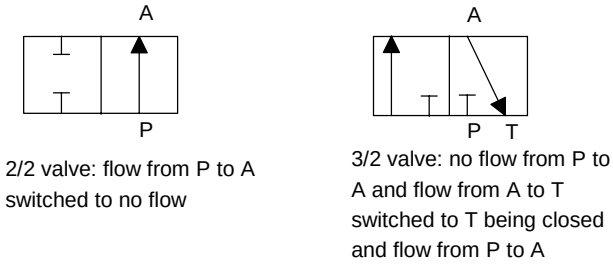

A
P
2/2 valve: flow from P to A
switched to no flow
A
P T
3/2 valve: no flow from P to
A and flow from A to T
switched to T being closed
and flow from P to A

Figure 2.26 *Direction valves*

In diagrams, the actuation methods used with valves are added to the symbol; Figure 2.27 shows examples of such symbols. The valve shown in Figure 2.24 has a spring to give one position and a solenoid to give the other and so the symbol is as shown in Figure 2.27(d).

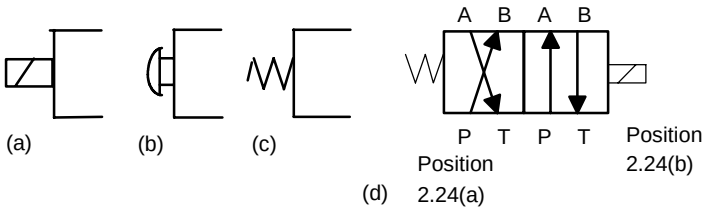


Figure 2.27 *Actuation symbols: (a) solenoid, (b) push button, (c) spring operated, (d) a 4/2 valve*

Direction valves can be used to control the direction of motion of pistons in cylinders, the displacement of the pistons being used to implement the required actions. The term *single acting cylinder* (Figure 2.28(a)) is used for one which is powered by the pressurised fluid being applied to one side of the piston to give motion in one direction, it being

returned in the other direction by possibly an internal spring. The term *double acting cylinder* (Figure 2.28(b)) is used when the cylinder is powered by fluid for its motion in both piston movement directions. Figure 2.29 shows how a valve can be used to control the direction of motion of a piston in a single-acting cylinder; Figure 2.30 shows how two valves can be used to control the action of a piston in a double acting cylinder.

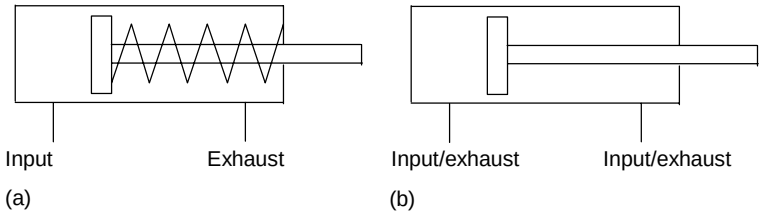


Figure 2.28 *Cylinders: (a) single acting, (b) double acting*

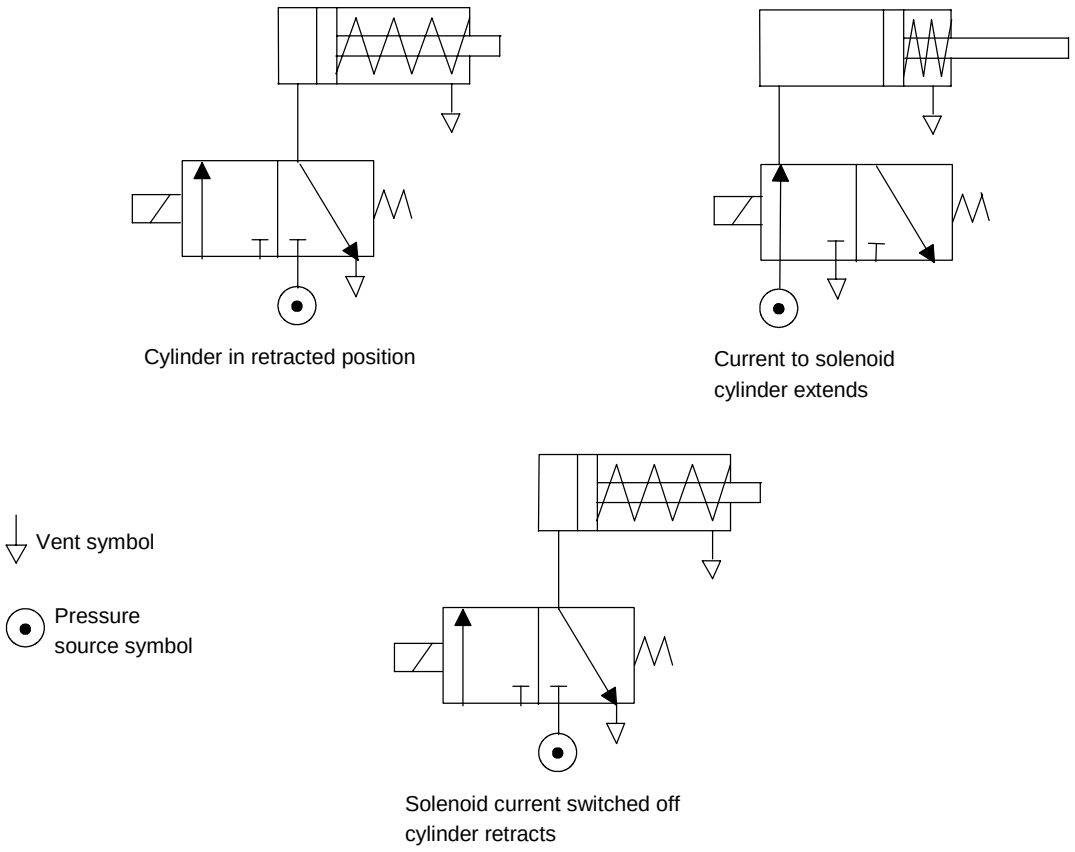


Figure 2.29 *Control of a single-acting cylinder*

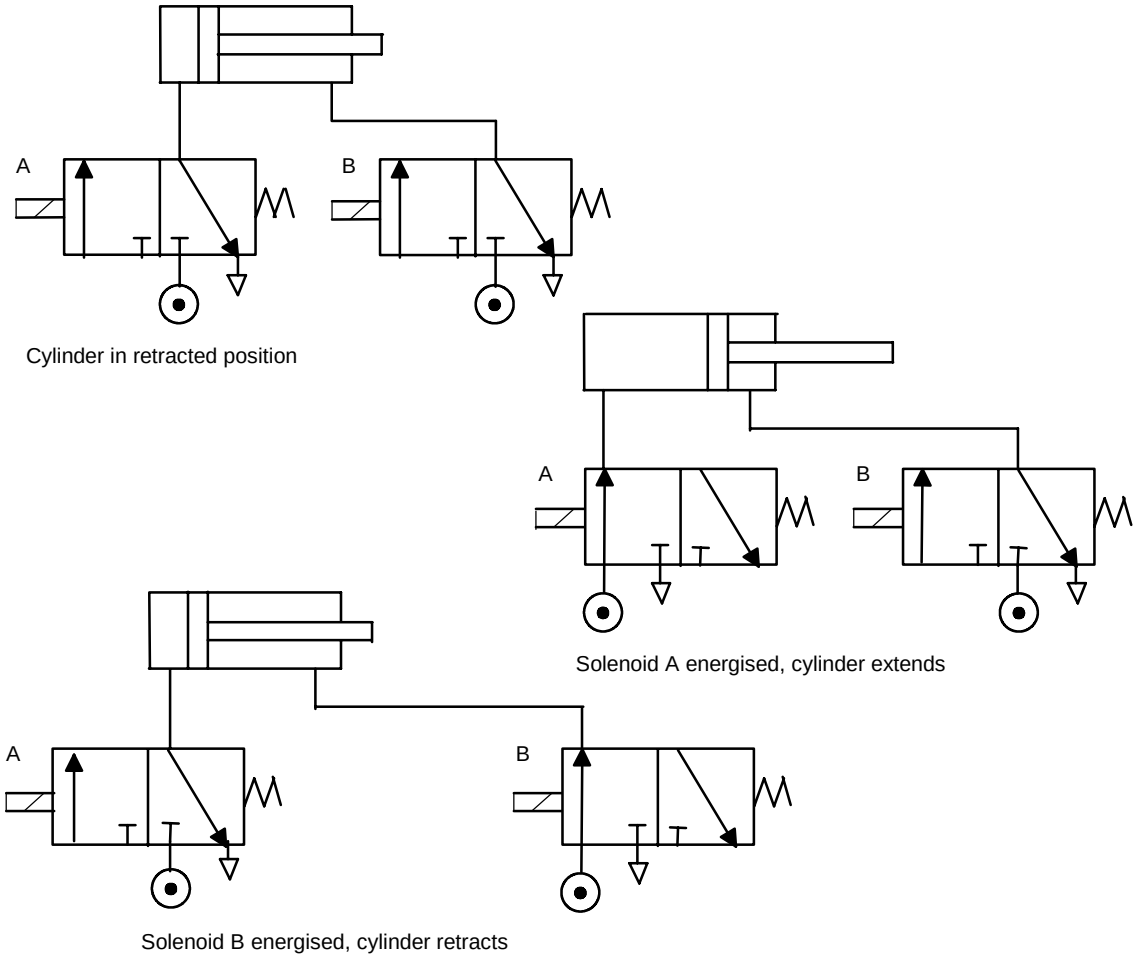


Figure 2.30 *Control of a double acting cylinder*

2.2.3 Motors

A *d.c. motor* has coils of wire mounted in slots on a cylinder of ferromagnetic material, this being termed the *armature*. The armature is mounted on bearings and is free to rotate. It is mounted in the magnetic field produced by permanent magnets or current passing through coils of wire, these being termed the *field coils*. When a current passes through the armature coil, forces act on the coil and result in rotation. Brushes and a commutator are used to reverse the current through the coil every half rotation and so keep the coil rotating. The speed of rotation can be changed by changing the size of the current to the armature coil. However, because fixed voltage supplies are generally used as the input to the coils, the required variable current is often obtained by an electronic circuit. This can control the average value of the voltage, and hence current, by varying the time for which the constant d.c. voltage is switched on (Figure 2.31). The term *pulse width modulation (PWM)* is used since the width of

the voltage pulses is used to control the average d.c. voltage applied to the armature. A PLC might thus control the speed of rotation of a motor by controlling the electronic circuit used to control the width of the voltage pulses.

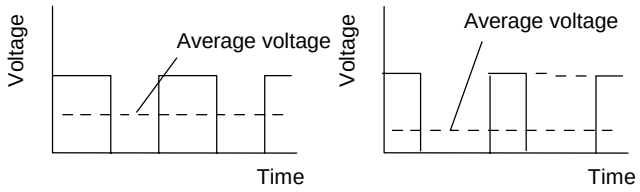


Figure 2.31 *Pulse width modulation*

Many industrial processes only require the PLC to switch a d.c. motor on or off. This might be done using a relay. Figure 2.32(a) shows the basic principle. The diode is included to dissipate the induced current resulting from the back e.m.f. Sometimes a PLC is required to reverse the direction of rotation of the motor. This can be done using relays to reverse the direction of the current applied to the armature coil. Figure 2.32(b) shows the basic principle. For rotation in one direction, switch 1 is closed and switch 2 opened. For rotation in the other direction, switch 1 is opened and switch 2 closed.

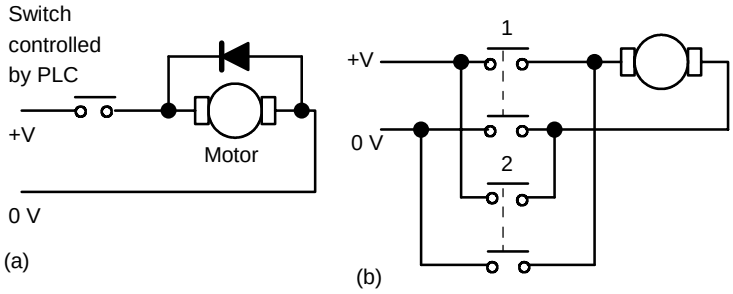


Figure 2.32 *D.c. motor: (a) on–off control, (b) directional control*

Another form of d.c. motor is the *brushless d.c. motor*. This uses a permanent magnet for the magnetic field but, instead of the armature coil rotating as a result of the magnetic field of the magnet, the permanent magnet rotates within the stationary coil. With the conventional d.c. motor, a commutator has to be used to reverse the current through the coil every half rotation in order to keep the coil rotating in the same direction. With the brushless permanent magnet motor, electronic circuitry is used to reverse the current. The motor can be started and stopped by controlling the current to the stationary coil. To reverse the motor, reversing the current is not so easy because of the electronic circuitry used for the commutator function. One method that is used is to incorporate sensors with the motor to detect the position of the north and south poles. These

sensors can then cause the current to the coils to be switched at just the right moment to reverse the forces applied to the magnet. The speed of rotation can be controlled using pulse width modulation, i.e. controlling the average value of pulses of a constant d.c. voltage.

Though a.c. motors are cheaper, more rugged and more reliable than d.c. motors, the maintaining of constant speed and controlling that speed is generally more complex than with d.c. motors. As a consequence, d.c. motors, particularly brushless permanent magnet motors, tend to be more widely used for control purposes.

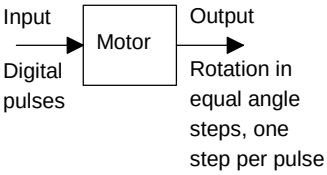


Figure 2.33 *The stepping motor*

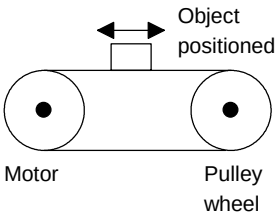


Figure 2.34 *Linear positioning*

2.2.4 Stepper motors

The *stepper* or *stepping motor* is a motor that produces rotation through equal angles, the so-termed *steps*, for each digital pulse supplied to its input (Figure 2.33). Thus, if one input pulse produces a rotation of 1.8° then 20 such pulses would give a rotation of 36.0°. To obtain one complete revolution through 360°, 200 digital pulses would be required. The motor can thus be used for accurate angular positioning.

If it is used to drive a continuous belt (Figure 2.34), it can be used to give accurate linear positioning. Such a motor is used with computer printers, robots, machine tools and a wide range of instruments where accurate positioning is required.

There are two basic forms of stepper motor, the *permanent magnet* type with a permanent magnet rotor and the *variable reluctance* type with a soft steel rotor. Figure 2.35 shows the basic elements of the permanent magnet type with two pairs of stator poles.

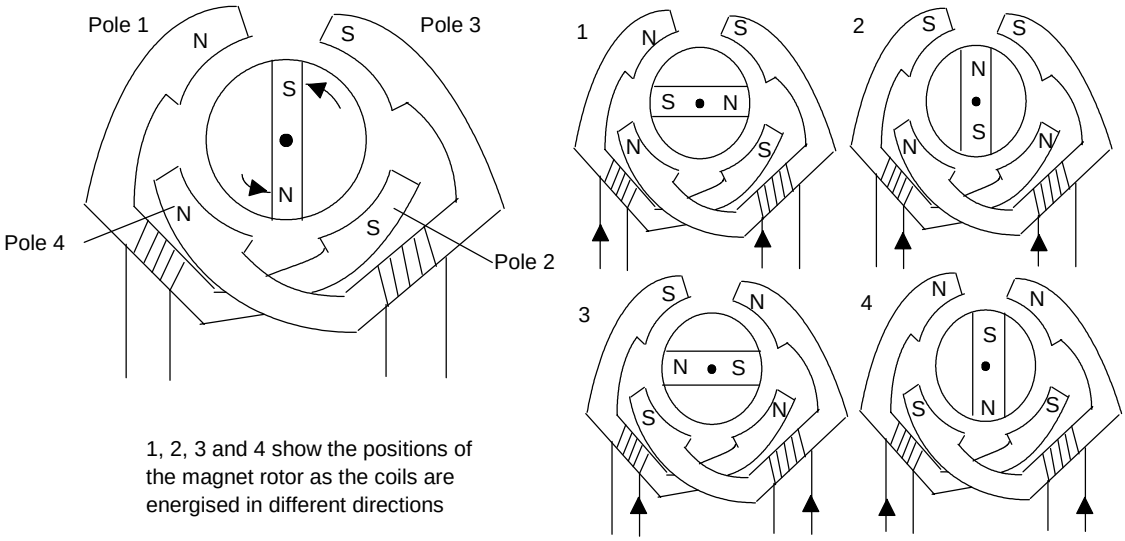


Figure 2.35 *The basic principles of the permanent magnet stepper motor (2-phase) with 90° steps*

Each pole is activated by a current being passed through the appropriate field winding, the coils being such that opposite poles are produced on opposite coils. The current is supplied from a d.c. source to the windings through switches. With the currents switched through the coils such that the poles are as shown in Figure 2.35, the rotor will move to line up with the next pair of poles and stop there. This would be, for Figure 6.35, an angle of 45°. If the current is then switched so that the polarities are reversed, the rotor will move a step to line up with the next pair of poles, at angle 135° and stop there. The polarities associated with each step are:

Step	Pole 1	Pole 2	Pole 3	Pole 4
1	North	South	South	North
2	South	North	South	North
3	South	North	North	South
4	North	South	North	South
5	Repeat of steps 1 to 4			

There are thus, in this case, four possible rotor positions: 45°, 135°, 225° and 315°.

Figure 2.36 shows the basic principle of the *variable reluctance* type. The rotor is made of soft steel and has a number of teeth, the number being less than the number of poles on the stator. The stator has pairs of poles, each pair of poles being activated and made into an electromagnet by a current being passed through the coils wrapped round them. When one pair of poles is activated, a magnetic field is produced which attracts the nearest pair of rotor teeth so that the teeth and poles line up. This is termed the position of *minimum reluctance*. By then switching the current to the next pair of poles, the rotor can be made to rotate to line up with those poles. Thus by sequentially switching the current from one pair of poles to the next, the rotor can be made to rotate in steps.

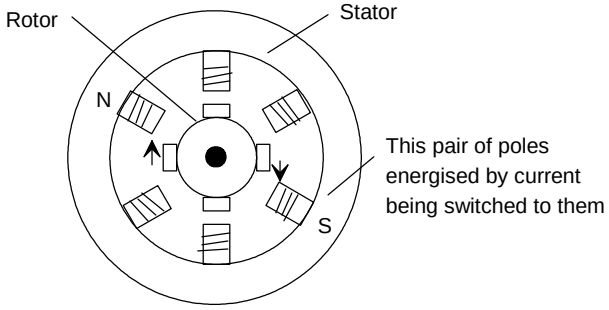


Figure 2.36 *The principle of the variable reluctance stepper motor*

There is another version of the stepper motor and that is a *hybrid stepper*. This combines features of both the permanent magnet and variable reluctance motors. They have a permanent magnet rotor encased

in iron caps which are cut to have teeth. The rotor sets itself in the minimum reluctance position in response to a pair of stator coils being energised.

To drive a stepper motor, so that it proceeds step-by-step to provide rotation, requires each pair of stator coils to be switched on and off in the required sequence when the input is a sequence of pulses (Figure 2.37). Driver circuits are available to give the correct sequencing and Figure 2.38 shows an example, the SAA 1027 for a four-phase unipolar stepper. Motors are termed *unipolar* if they are wired so that the current can only flow in one direction through any particular motor terminal, *bipolar* if the current can flow in either direction through any particular motor terminal. The stepper motor will rotate through one step each time the trigger input goes from low to high. The motor runs clockwise when the rotation input is low and anticlockwise when high. When the set pin is made low the output resets. In a control system, these input pulses might be supplied by a microprocessor.

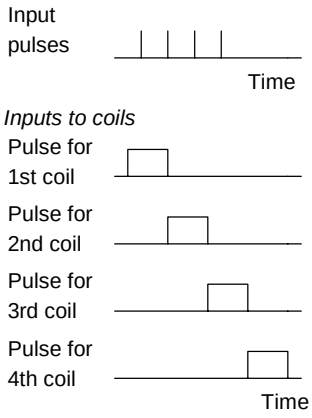


Figure 2.37 *Input and outputs of the drive system*

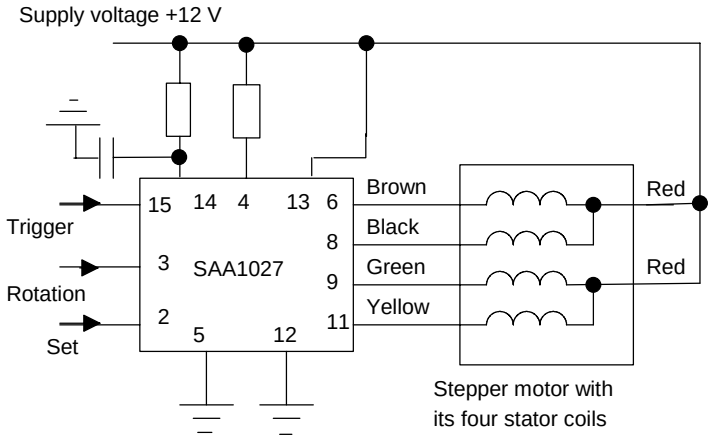


Figure 2.38 *Driver circuit connections with the integrated circuit SAA1027*

2.3 Examples of applications

The following are some examples of control systems designed to illustrate the use of a range of input and output devices.

2.3.1 A conveyor belt

Consider a conveyor belt that is to be used to transport goods from a loading machine to a packaging area (Figure 2.39). When an item is loaded onto the conveyor belt, a contact switch might be used to indicate that the item is on the belt and start the conveyor motor. The motor then has to keep running until the item reaches the far end of the conveyor and falls off into the packaging area. When it does this, a switch might be activated which has the effect of switching off the conveyor motor. The motor is then to remain off until the next item is loaded onto the belt. Thus the inputs to a PLC controlling the conveyor are from two switches and the output is to a motor.

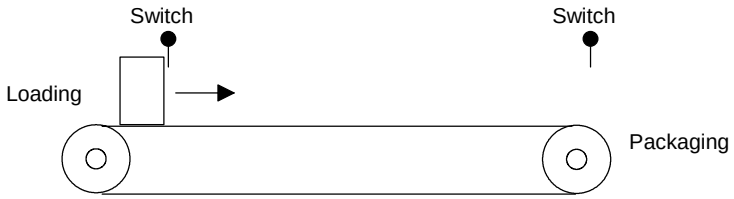


Figure 2.39 *Conveyor*

2.3.2 A lift

Consider a simple goods lift to move items from one level to another. It might be bricks from the ground level to the height where the bricklayers are working. The lift is to move upwards when a push button is pressed at the ground level to send the lift upwards or a push button is pressed at the upper level to request the lift to move upwards, but in both cases there is a condition that has to be met that a limit switch indicates that the access gate to the lift platform is closed. The lift is to move downwards when a push button is pressed at the upper level to send the lift downwards or a push button is pressed at the lower level to request the lift to move downwards, but in both cases there is a condition that has to be met that a limit switch indicates that the access gate to the lift platform is closed. Thus the inputs to the control system are electrical on–off signals from push button switches and limit switches. The output from the control system is the signal to control the motor.

2.3.3 A robot control system

Figure 6.40 shows how directional control valves can be used for a control system of a robot. When there is an input to solenoid A of valve 1, the piston moves to the right and causes the gripper to close. If solenoid B is energised, with A de-energised, the piston moves to the left and the gripper opens. When both solenoids are de-energised, no air passes to either side of the piston in the cylinder and the piston keeps its position without change. Likewise, inputs to the solenoids of valve 2 are used to extend or retract the arm. Inputs to the solenoids of valve 3 are used to

move the arm up or down. Inputs to the solenoids of valve 4 are used to rotate the base in either a clockwise or anticlockwise direction.

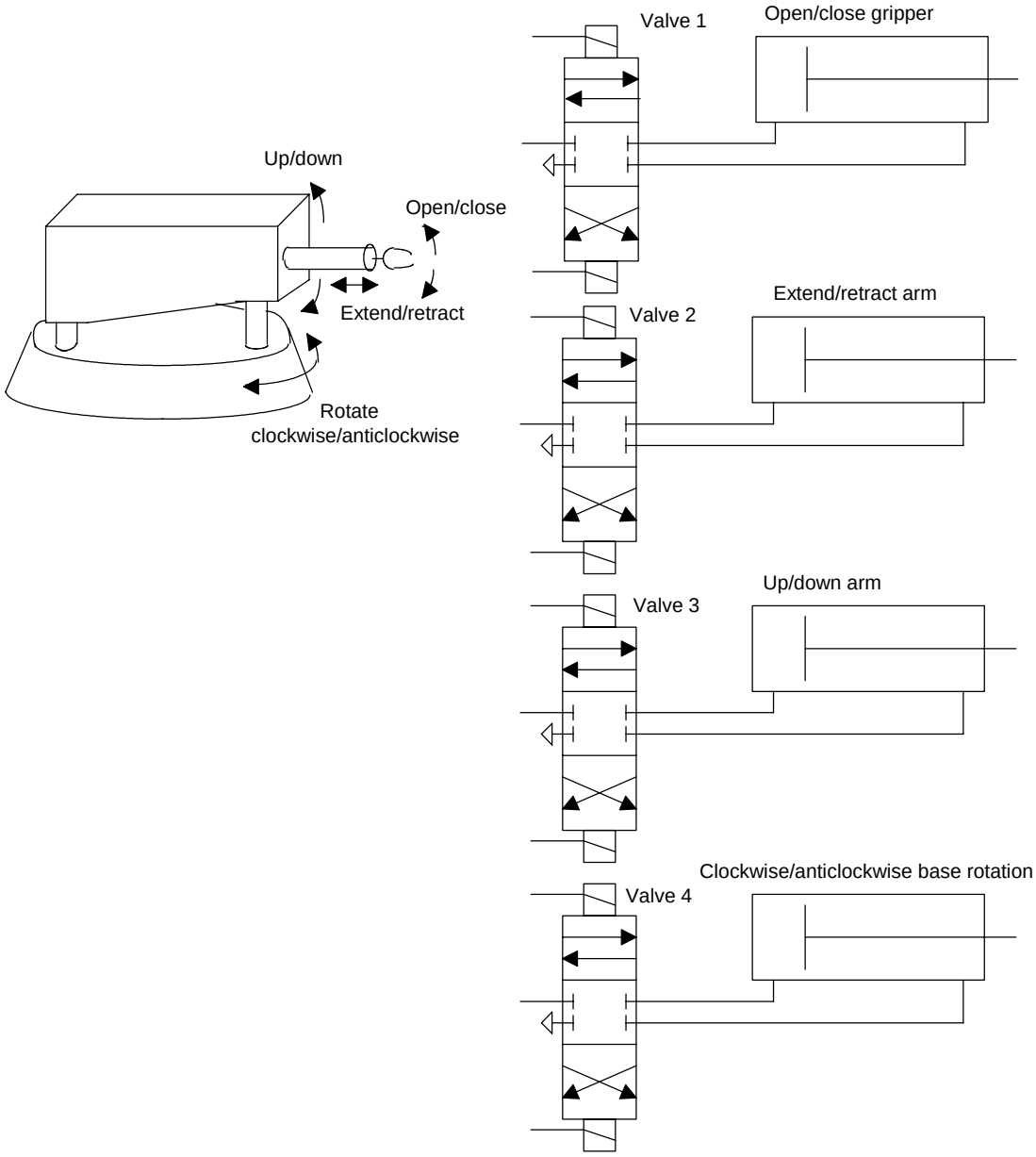


Figure 2.40 *Robot controls*

2.3.4 Liquid level monitoring

Figure 6.41 shows a method that could be used to give an on–off signal when the liquid in a container reaches a critical level. A magnetic float, a ring circling the sensor probe, falls as the liquid level falls and opens a

reed switch when the critical level is reached. The reed switch is in series with a 39 Ω resistor so that this is switched in parallel with a 1 kΩ resistor by the action of the reed switch. Opening the reed switch thus increases the resistance from about 37 Ω to 1 kΩ. Such a resistance change can be transformed by signal conditioning to give suitable on–off signals.

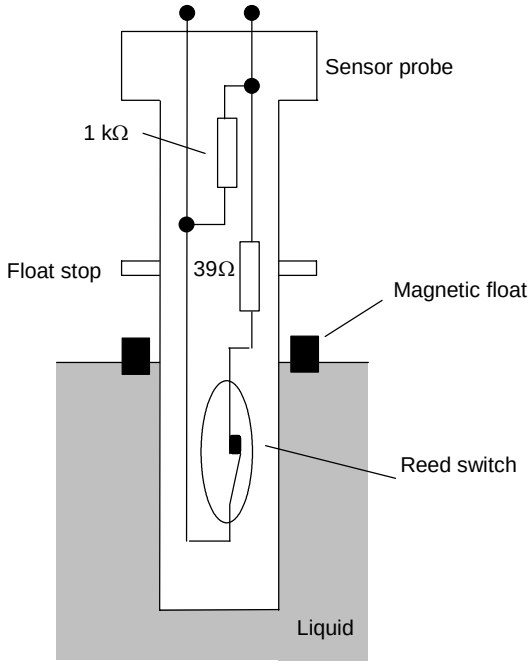


Figure 2.41 *Liquid level monitoring*

Problems

Questions 1 to 10 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

1 Decide whether each of these statements is True (T) or False (F).

A limit switch:
(i) Can be used to detect the presence of a moving part.
(ii) Is activated by contacts making or breaking an electrical circuit.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

2 Decide whether each of these statements is True (T) or False (F).

A thermistor is a temperature sensor which gives resistance changes which are:
(i) A non-linear function of temperature.
(ii) Large for comparatively small temperature changes.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

3 A diaphragm pressure sensor is required to give a measure of the gauge pressure present in a system. Such a sensor will need to have a diaphragm with:

A A vacuum on one side.
B One side open to the atmosphere.
C The pressure applied to both sides.
D A controlled adjustable pressure applied to one side.

4 The change in resistance of an electrical resistance strain gauge with a gauge factor of 2.0 and resistance 100 Ω when subject to a strain of 0.001 is:

A 0.0002 Ω
B 0.002 Ω
C 0.02 Ω
D 0.2 Ω

5 An incremental shaft encoder gives an output which is a direct measure of:

A The diameter of the shaft.
B The change in diameter of the shaft.
C The change in angular position of the shaft.
D The absolute angular position of the shaft.

6 Decide whether each of these statements is True (T) or False (F).

Input devices which give an analogue input for displacement are:
(i) Linear potentiometer.
(ii) Linear variable differential transformer.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 7 and 8 refer to Figure 2.42 which shows the symbol for a directional valve.

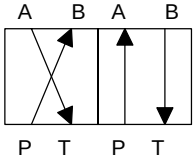


Figure 2.42 *Problems 7 and 8*

7 Decide whether each of these statements is True (T) or False (F).

The valve has:
(i) 4 ports
(ii) 2 positions

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

8 Decide whether each of these statements is True (T) or False (F).

In the control positions:
(i) A is connected to T and P to B.
(ii) P is connected to A and B to T.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

9 Decide whether each of these statements is True (T) or False (F).

A stepper motor has a step angle of 1.8°. This means that:
(i) Each pulse input to the motor rotates the motor shaft by 1.8°.
(ii) The motor shaft takes 1 s to rotate through 1.8°.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

10 Decide whether each of these statements is True (T) or False (F).

A proximity switch is required for detecting the presence of a non-metallic object. Types of switches that might be suitable are:
(i) Eddy current type.
(ii) Capacitive type.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

11 Explain the operation of the following input devices, stating the form of the signal being sensed and the output: (a) reed switch, (b) incremental shaft encoder, (c) photoelectric transmissive switch, (d) diaphragm pressure switch.

12 Explain how the on–off operation and direction of a d.c. motor can be controlled by switches.

13 Explain the principle of the stepper motor.

3 Number systems

The number system used for everyday calculations is the *denary* or *decimal system*. This is based on the use of the 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. With a number represented by this system, the digit position in the number indicates the weight attached to each digit, the weight increasing by a factor of 10 as we proceed from right to left. Hence we have:

...	10^3	10^2	10^1	10^0
	thousands	hundreds	tens	units
Denary	1000	100	10	1

Counting can, however, be done to any base. The denary system is just convenient because we have ten fingers. If we had only two then our system for everyday counting would probably have been different. Computers, and hence PLC systems, are based on counting in twos because it is convenient for their system, their two digits being effectively just the off and on signals. When working with PLCs, other base number systems are also used, e.g. input and output addresses are often specified using the octal system, i.e. base 8.

3.1 The binary system

The *binary system* is based on just two digits: 0 and 1. These are termed *b*inary digi*ts* or *bits*. When a number is represented by this system, the digit position in the number indicates the weight attached to each digit, the weight increasing by a factor of 2 as we proceed from right to left.

...	2^3	2^2	2^1	2^0
	bit 3	bit 2	bit 1	bit 0
Binary	1000	100	10	1

The bit 0 is termed the *least significant bit* (LSB) and the highest bit the *most significant bit* (MSB). For example, with the binary number 1010, the least significant bit is the bit at the right-hand end of the number and so is 0. The most significant bit is the bit at the left-hand end of the number and so is 1.

	2^3	2^2	2^1	2^0
	bit 3	bit 2	bit 1	bit 0
	MSB			LSB
Binary	1	0	1	0

When converted to a denary number we have, for the 1010:

	2^3	2^2	2^1	2^0
Binary	1	0	1	0
Denary	$2^3 = 8$	0	$2^1 = 2$	0

Thus the denary equivalent is 10. The conversion of a binary number to a denary number thus involves the addition of the powers of 2 indicated by the number.

The conversion of a denary number to a binary number involves looking for the appropriate powers of 2. We can do this by successive divisions by 2, noting the remainders at each division. Thus with the denary number 31:

31 ÷ 2 = 15 remainder 1 This gives the LSB
15 ÷ 2 = 7 remainder 1
7 ÷ 2 = 3 remainder 1
3 ÷ 2 = 1 remainder 1 This gives the MSB

The binary number is 11111. The first division gives the least significant bit because we have just divided the 31 by 2, i.e. 2^1 and found 1 left over for the 2^0 digit. The last division gives the most significant bit because the 31 has then been divided by 2 four times, i.e. 2^4 and the remainder is 1.

3.2 Octal and hexadecimal

Binary numbers are used in computers because the two states represented by 0 and 1 are easy to deal with switching circuits where they can represent off and on. A problem with binary numbers is that a comparatively small number requires a large number of digits. For example, the denary number 9 which involves just a single digit requires four when written as the binary number 1001. The denary number 181, involving three digits, in binary form is 10110101 and requires eight digits. Because of this, octal or hexadecimal numbers are sometimes used to make numbers easier to handle and act as a ‘half-way house’ between denary numbers and the binary numbers which computers work with.

3.2.1 Octal system

The *octal system* is based on eight digits: 0, 1, 2, 3, 4, 5, 6, 7. When a number is represented by this system, the digit position in the number indicates the weight attached to each digit, the weighting increasing by a factor of 8 as we proceed from right to left. Thus we have:

...	8^3	8^2	8^1	8^0
Octal	1000	100	10	1

To convert denary numbers to octal we successively divide by 8 and note the remainders. Thus the denary number 15 divided by 8 gives 1 with remainder 7 and thus the denary number 15 is 17 in the octal system. To convert from octal to denary we multiply the digits by the power of 8 appropriate to its position in the number. For example, the octal number 365 is $3 \times 8^2 + 6 \times 8^1 + 5 \times 8^0 = 245$. To convert from binary into octal, the binary number is written in groups of three bits starting with the least

significant bit. For example, the binary number 11010110 would be written as:

```
11  010  110
```

Each group is then replaced by the corresponding digit 0 to 7. The 110 binary number is 6, the 010 is 2 and the 11 is 3. Thus the octal number is 326. As another example, the binary number 100111010 is:

```
100 111 010   Binary
 4   7   2    Octal
```

Octal to binary conversion involves converting each octal digit into its 3-bit equivalent. Thus, for the octal number 21 we have 1 as 001 and 2 as 010:

```
 2   1     Octal number
010 001   Binary number
```

and so the binary number is 010001.

3.2.2 Hexadecimal system

The *hexadecimal system (hex)* is based on 16 digits/symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. When a number is represented by this system, the digit position in the number indicates that the weight attached to each digit increases by a factor of 16 as we proceed from right to left. Thus we have:

...	16^3	16^2	16^1	16^0
Hex	1000	100	10	1

For example, the decimal number 15 is F in the hexadecimal system. To convert from denary numbers into hex we successively divide by 16 and note the remainders. Thus the denary number 156 when divided by 16 gives 9 with remainder 12 and so in hex is 9C. To convert from hex to denary we multiply the digits by the power of 16 appropriate to its position in the number. Thus hex 12 is $1 \times 16^1 + 2 \times 16^0 = 18$. To convert binary numbers into hexadecimal numbers, we group the binary numbers into fours starting from the least significant number. Thus, for the binary number 1110100110 we have:

```
11  1010  0110  Binary number
3     A     6   Hex number
```

For conversion from hex to binary, each hex number is converted to its 4-bit equivalent. Thus, for the hex number 1D we have 0001 for the 1 and 1101 for the D:

```
 1     D    Hex number
0001 1101  Binary number
```

Thus the binary number is 0001 1101.

Because the external world tends to deal mainly with numbers in the denary system and computers with numbers in the binary system, there is always the problem of conversion. There is, however, no simple link between the position of digits in a denary number and the position of digits in a binary number. An alternative method that is often used is the *binary coded decimal system (BCD)*. With this system, each denary digit is coded separately in binary. For example, the denary number 15 has the 5 converted into the binary number 0101 and the 1 into 0001:

```
   1      5   Denary number
0001 0101   Binary number
```

to give in BCD the number 0001 0101.

3.2.3 Numbers in the binary, octal, hex and BCD systems

Table 3.1 gives examples of numbers in the denary, binary, octal, hex and BCD systems.

Table 3.1 *Examples of numbers in different systems*

Denary	Binary	Octal	Hex	BCD
0	00000	0	0	0000 0000
1	00001	1	1	0000 0001
2	00010	2	2	0000 0010
3	00011	3	3	0000 0011
4	00100	4	4	0000 0100
5	00101	5	5	0000 0101
6	00110	6	6	0000 0110
7	00111	7	7	0000 0111
8	01000	10	8	0000 1000
9	01001	11	9	0000 1001
10	01010	12	A	0001 0000
11	01011	13	B	0001 0001
12	01100	14	C	0001 0010
13	01101	15	D	0001 0011
14	01110	16	E	0001 0100
15	01111	17	F	0001 0101
16	10000	20	10	0001 0110
17	10001	21	11	0001 0111

3.3 Binary arithmetic

Addition of binary numbers uses the following rules:

$0 + 0 = 0$

$0 + 1 = 1 + 0 = 1$

1 + 1 = 10

1 + 1 + 1 = 11

Consider the addition of the binary numbers 01110 and 10011.

	01110
	10011
Sum	100001

For bit 0 in the sum, 0 + 1 = 1. For bit 1 in the sum, 1 + 1 = 10 and so we have 0 with 1 carried to the next column. For bit 2 in the sum, 1 + 0 + the carried 1 = 10. For bit 3 in the sum, 1 + 0 + the carried 1 = 10. We continue this through the various bits and end up with the 100001.

Subtraction of binary numbers follows the following rules:

0 – 0 = 0

1 – 0 = 1

1 – 1 = 0

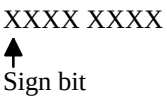
When evaluating 0 – 1, a 1 is borrowed from the next column on the left containing a 1. The following example illustrates this with the subtraction of 01110 from 11011:

	11011
	01110
Difference	01101

For bit 0 we have 1 – 0 = 1. For bit 1 we have 1 – 1 = 0. For bit 2 we have 0 – 1. We thus borrow 1 from the next column and so have 10 – 1 = 1. For bit 3 we have 0 – 1, remember we borrowed the 1. Again borrowing 1 from the next column, we then have 10 – 1 = 1. For bit 4 we have 0 – 0 = 0, remember we borrowed the 1.

3.3.1 Signed numbers

The binary numbers considered so far contain no indication whether they are negative or positive and are said to be *unsigned*. Since there is generally a need to handle both positive and negative numbers there needs to be some way of distinguishing between them. This can be done by adding a sign bit. When a number is said to be *signed* then the most significant bit is used to indicate the sign of the number, a 0 being used if the number is positive and a 1 if it is negative. Thus for an 8-bit number we have:



When we have a positive number then we write it in the normal way with a 0 preceding it. Thus a positive binary number of 10110 would be written as 010110. A negative number of 10110 would be written as 110110. However, this is not the most useful way of writing negative numbers for ease of manipulation by computers.

3.3.2 One's and two's complements

A more useful way of writing signed negative numbers is to use the two's complement method. A binary number has two complements, known as the *one's complement* and the *two's complement*. The one's complement of a binary number is obtained by changing all the 1s in the unsigned number into 0s and the 0s into 1s. Thus if we have the binary number 101101 then the one's complement of it is 010010. The two's complement is obtained from the one's complement by adding 1 to the least significant bit of the one's complement. Thus the one's complement of 010010 becomes 010011.

When we have a negative number then, to obtain the signed two's complement, we obtain the two's complement and then sign it with a 1. Consider the representation of the decimal number –6 as a signed two's complement number when the total number of bits is to be eight. We first write the binary number for +6, i.e. 0000110, then obtain the one's complement of 1111001, add 1 to give 1111010, and finally sign it with a 1 to indicate it is negative. The result is thus 11111010.

Unsigned binary number when sign ignored	000 0110
One's complement	111 1001
Add 1	1
Unsigned two's complement	111 1010
Signed two's complement	1111 1010

Table 3.2 lists some signed two's complements, given to 4 bits, for denary numbers.

When we have a positive number then we sign the normal binary number with a 0, i.e. we only write negative numbers in the two's complement form. A consequence of adopting this method of writing negative and positive numbers is that when we add the signed binary equivalent of +4 and –4, i.e. 0000 0100 and 111 1100 we obtain (1)0000 0000 and so zero within the constraints of the number of bits used, the (1) being neglected.

Table 3.2 *Signed two's complements*

Denary number	Signed 2's complement
–5	1011
–4	1100
–3	1101
–2	1110
–1	1111

Subtraction of a positive number from a positive number can be considered to be the addition of a negative number to a positive number. Thus we obtain the signed two's complement of the negative number and then add it to the signed positive number. Hence, for the subtraction of the denary number 6 from the denary number 4 we can consider the problem as being (+4) + (–6). Hence we add the signed positive number to the signed two's complement for the negative number.

Binary form of +4	0000 0100
(–6) as signed two's complement	1111 1010
Sum	1111 1110

The most significant bit, i.e. the sign, of the outcome is 1 and so the result is negative. This is the 8-bit signed two's complement for –2.

If we wanted to add two negative numbers then we would obtain the signed two's complement for each number and then add them. Whenever a number is negative we use the signed two's complement, when positive just the signed number.

3.3.3 Floating point numbers

Before discussing floating point numbers, consider *fixed point numbers*. Fixed point numbers are ones where there is a fixed location of the point separating integers from fractional numbers. Thus, 15.3 is an example of a denary fixed point number, 1010.1100 an example of a fixed point binary number and DE.2A an example of a fixed point hexadecimal number. We have, with the eight-bit binary number, four digits before the binary point and four digits after it. When two such binary numbers are added by a computing system the procedure is to recognise that the fixed point is fixed the same in both numbers so we can ignore it for the addition, carry out the addition of the numbers and then insert in the result the binary point in its fixed position. For example, suppose we want to add 0011.1010 and 0110.1000, we drop the binary point to give:

0011 1010 + 0110 1000 = 1010 0010

Inserting the binary point then gives 1010.0010.

Using fixed points does present problems. If we are concerned with very large or very small numbers we could end up with a large number of zeros between the integers and the point, e.g. 0.000 000 000 000 023. For

this reason *scientific notation* is used for such numbers. Thus, the above number might be written as 0.23×10^{-13} or 2.3×10^{-14} or 23×10^{-15}. Likewise, the binary number 0.0000 0111 0010 might be represented as 110010×2^{-12} (the 12 would also be in binary format) or 11001.0×2^{-11} (the 11 being in binary format). Such notation has a *floating point*.

A floating point number is in the form $a \times r^e$, where a is termed the *mantissa*, r the *radix* or *base* and e the *exponent* or *power*. With binary numbers the base is understood to be 2, i.e. we have $a \times 2^e$, and when we know we are dealing with binary numbers we need not store the base with the number. Thus a computing system needs, in addition to storing the sign, i.e. whether positive or negative, to store the mantissa and the exponent.

Because with floating point numbers it is possible to store a number in several different ways, e.g. 0.1×10^2 and 0.01×10^3, with computing systems such numbers are normalised. This means that they are all put in the form $0.1 \times r^e$. Thus, with binary numbers we have 0.1×2^e, e.g. if we had 0.00001001 it would become 0.1001×2^{-4}. In order to take account of the sign of a binary number we then add a sign bit of 0 for a positive number and 1 for a negative number. Thus the number 0.1001×2^{-4} becomes 1.1001×2^{-4} if negative and 0.1001×2^{-4} if positive.

Unlike fixed point numbers, floating point numbers cannot be directly added unless the exponents are the same. Thus to carry out addition we need to make the exponents the same.

3.4 PLC data

Most PLCs operate with a 16-bit word, the term *word* being used for the group of bits constituting some information. This allows a positive number in the range 0 to +65 535, i.e. 1111 1111 1111 1111, to be represented or a signed number in the range –32 768 to +32 767 in two's complement, the most significant bit then representing the sign. Such signed numbers are referred to as integers with the symbol INT being used with inputs and outputs in programs of such 16-bit words. The term SINT is used for short integer numbers where only 8-bits are used, such numbers giving the range –128 to +127. The term DINT is used for double integer numbers where 32 bits are used, such numbers giving the range -2^{31} to $+2^{31} - 1$. LINT is used for long integer numbers where 64 bits are used, such numbers giving the range -2^{63} to $+2^{63} - 1$. Where numbers are not signed the symbols UINT, USINT, UDINT and ULINT are used with integers, short integers, double integers and long integers.

Decimal fractions are referred to as real or floating point numbers, being represented by the symbol REAL for inputs and outputs in programs. These consist of two 16-bit words and so we might have 1.234567E+03 for the number $1.234\,567 \times 10^{+3}$, the E indicating that the number that follows is the exponent. The term LREAL is used for long real numbers where 64 bits are used.

The term BOOL is used for Boolean type data, such data being on/off values, i.e. 0 or 1, and thus represented by single bits.

Time duration, e.g. for the duration of a process, is represented by the IEC (International Electrotechnical Commission) standard using the symbols d for days, h for hours, m for minutes, s for seconds and ms for milliseconds as, for example, T#12d2h5s3ms or TIME#12d2h5s for 12

days 2 hours 5 seconds and 3 milliseconds. Note that # is the symbol used to indicate that which follows is a numerical quantity.

Problems

1 Convert the following binary numbers to denary numbers: (a) 000011, (b) 111111, (c) 001101.

2 Convert the following denary numbers to binary numbers: (a) 100, (b) 146, (c) 255.

3 Convert the following hexadecimal numbers to denary numbers: (a) 9F, (b) D53, (c) 67C.

4 Convert the following denary numbers to hexadecimal numbers: (a) 14, (b) 81, (c) 2562.

5 Convert the following hexadecimal numbers to binary numbers: (a) E, (b) 1D, (c) A65.

6 Convert the following octal numbers to denary numbers: (a) 372, (b) 14, (c) 2540.

7 Convert the following denary numbers to octal numbers: (a) 20, (b) 265, (c) 400.

8 Convert the following octal numbers to binary numbers: (a) 270, (b) 102, (c) 673.

9 Convert the following decimal numbers to BCD equivalents: (a) 20, (b) 35, (c) 92.

10 Convert the following denary numbers to signed two's complement binary 8-bit format: (a) –1, (b) –35, (c) –125.

11 Convert the following signed two's complement binary 8-bit numbers to their denary equivalents: (a) 1111 0000, (b) 1100 1001, (c) 1101 1000.

12 Convert the following binary numbers to normalised floating point numbers: (a) 0011 0010, (b) 0000 1100, (c) 1000.0100.

4 I/O processing

This chapter continues the discussion of inputs and outputs from Chapter 2 and is a brief consideration of the processing of the signals from input and output devices. The input/output (I/O) unit provides the interface between the PLC controller and the outside world and must therefore provide the necessary signal conditioning to get the signal to the required level and also to isolate it from possible electrical hazards such as high voltages. This chapter includes the forms of typical input/output modules and, in an installation where sensors are some distance from the PLC processing, their communication links to the PLC.

4.1 Input/output units

Input signals from sensors and the outputs required for actuating devices can be:

1 *Analogue*, i.e. a signal whose size is related to the size of the quantity being sensed.
2 *Discrete*, i.e. essentially just an on–off signal.
3 *Digital*, i.e. a sequence of pulses.

The CPU, however, must have an input of digital signals of a particular size, normally 0 to 5 V. The output from the CPU is digital, normally 0 to 5 V. Thus there is generally a need to manipulate input and output signals so that they are in the required form.

The input/output (I/O) units of PLCs are designed so that a range of input signals can be changed into 5 V digital signals and so that a range of outputs are available to drive external devices. It is this in-built facility to enable a range of inputs and outputs to be handled which makes PLCs so easy to use. The following is a brief indication of the basic circuits used for input and output units. In the case of rack instruments they are mounted on cards which can be plugged into the racks and so the input/output characteristics of the PLC can thus be changed by changing the cards. A single box form of PLC has input/output units incorporated by the manufacturer.

4.1.1 Input units

The terms *sourcing* and *sinking* refer to the manner in which d.c. devices are interfaced with the PLC (see Section 1.3.5). For a PLC input unit, with sourcing it is the source of the current supply for the input device connected to it (Figure 4.1(a)). With sinking, the input device provides the current to the input unit (Figure 4.1(b)).

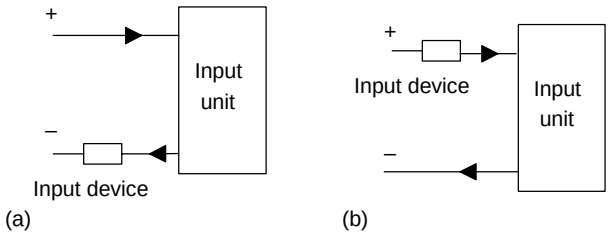


Figure 4.1 *Input unit: (a) sourcing, (b) sinking*

Figures 4.2 and 4.3 show the basic input unit circuits for discrete and digital d.c. and discrete a.c. inputs. Optoisolators (see Section 1.3.4) are used to provide protection. With the a.c. input unit, a rectifier bridge network is used to rectify the a.c. so that the resulting d.c. signal can provide the signal for use by the optoisolator to give the input signals to the CPU of the PLC. Individual status lights are provided for each input to indicate when the input device is providing a signal.

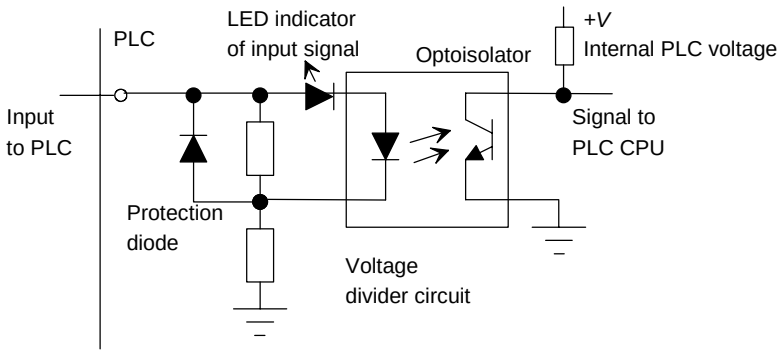


Figure 4.2 *D.C. input unit*

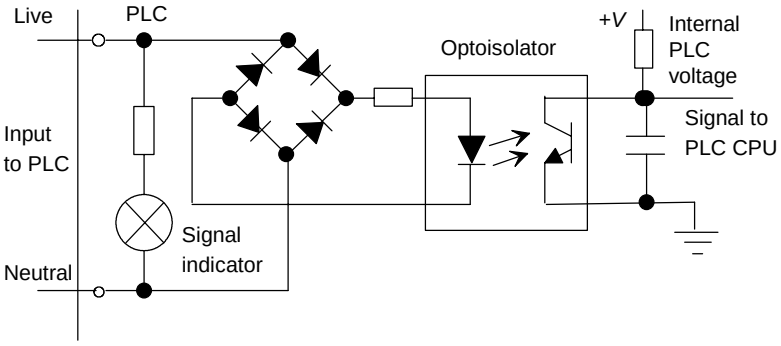


Figure 4.3 *A.C. input unit*

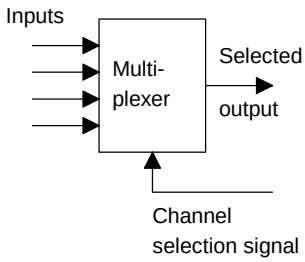


Figure 4.4 *Multiplexer*

Analogue signals can be inputted to a PLC if the input channel is able to convert the signal to a digital signal using an analogue-to-digital converter. With a rack mounted system this may be achieved by mounting a suitable analogue input card in the rack. So that one analogue card is not required for each analogue input, multiplexing is generally used (Figure 4.4). This involves more than one analogue input being connected to the card and then electronic switches used to select each input in turn. Cards are typically available giving 4, 8 or 16 analogue inputs.

Figure 4.5(a) illustrates the function of an analogue-to-digital converter (ADC). A single analogue input signal gives rise to on–off output signals along perhaps eight separate wires. The eight signals then constitute the so-termed digital *word* corresponding to the analogue input signal level. With such an 8-bit converter there are $2^8 = 256$ different digital values possible; these are 0000 0000 to 1111 1111, i.e. 0 to 255. The digital output goes up in steps (Figure 4.5(b)) and the analogue voltages required to produce each digital output are termed *quantisation levels*.

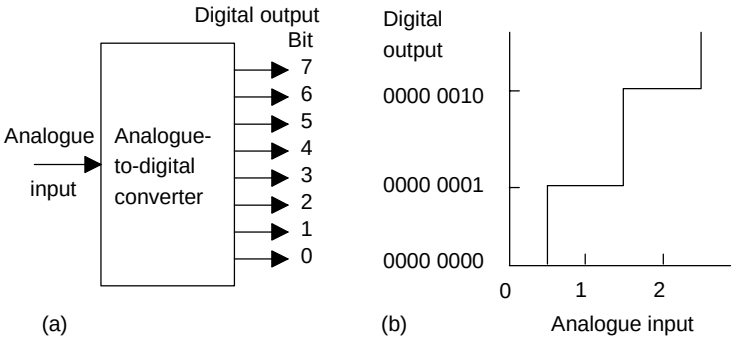


Figure 4.5 *(a) Function of an analogue-to-digital converter, (b) an illustration of the relationship between the analogue input and the digital output*

The analogue voltage has to change by the difference in analogue voltage between successive levels if the binary output is to change. The term *resolution* is used for the smallest change in analogue voltage which will give rise to a change in one bit in the digital output. With an 8-bit ADC, if, say, the full-scale analogue input signal varies between 0 and 10 V then a step of one digital bit involves an analogue input change of 10/255 V or about 0.04 V. This means that a change of 0.03 V in the input will produce no change in the digital output. The number of bits in the output from an analogue-to-digital converter thus determines the *resolution*, and hence *accuracy*, possible. If a 10-bit ADC is used then $2^{10} = 1024$ different digital values are possible and, for the full-scale analogue input of 0 to 10 V, a step of one digital bit involves an analogue input change of 10/1023 V or about 0.01 V. If a 12-bit ADC is used then $2^{12} = 4096$ different digital values are possible and, for the full-scale analogue input of 0 to 10 V, a step of one digital bit involves an analogue input change of 10/4095 V or about 2.4 mV. In general, the resolution of an n-bit ADC is $1/(2^n - 1)$.

The following illustrates the analogue-to-digital conversion for an 8-bit converter when the analogue input is in the range 0 to 10 V:

Analogue input (V)	Digital output (V)
0.00	0000 0000
0.04	0000 0001
0.08	0000 0010
0.12	0000 0011
0.16	0000 0100
0.20	0000 0101
0.24	0000 0110
0.28	0000 0111
0.32	0000 1000
etc.	

To illustrate the above, consider a thermocouple used as a sensor with a PLC and giving an output of 0.5 mV per °C. What will be the accuracy with which the PLC will activate the output device if the thermocouple is connected to an analogue input with a range of 0 to 10 V d.c and using a 10-bit analogue-to-digital converter? With a 10-bit converter there is 2^{10} = 1024 bits covering the 0 to 10 V range. Thus a change of 1 bit corresponds to 10/1023 V or about 0.01 V, i.e. 10 mV. Hence the accuracy with which the PLC recognises the input from the thermocouple is ±5 mV or ±10°C.

4.1.2 Output units

With a PLC output unit, when it provides the current for the output device (Figure 4.6(a)) it is said to be sourcing and when the output device provides the current to the output unit it is said to be sinking (Figure 4.6(b)). Quite often, sinking input units are used for interfacing with electronic equipment and sourcing output units for interfacing with solenoids.

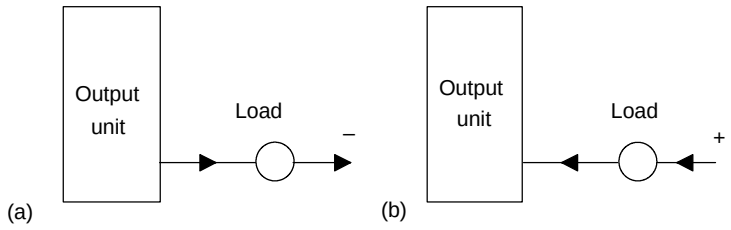


Figure 4.6 *Output unit: (a) sourcing, (b) sinking*

Output units can be relay, transistor or triac. Figure 4.7 shows the basic form of a relay output unit, Figure 4.8 that of a transistor output unit and Figure 4.9 that of a triac output unit.

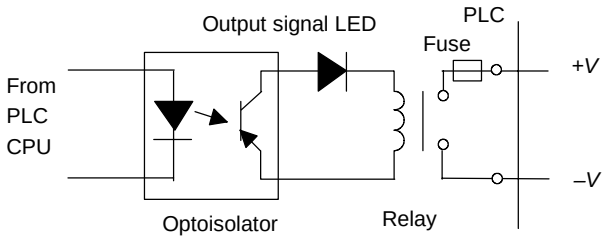


Figure 4.7 *Relay output unit*

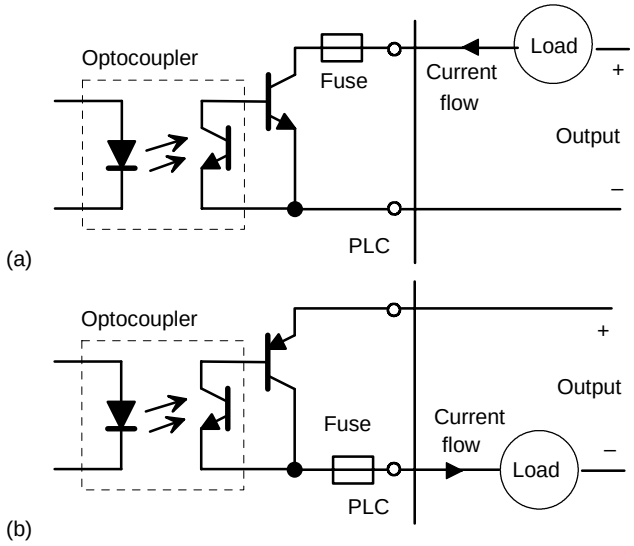


Figure 4.8 *Basic forms of transistor output: (a) current sinking, (b) current sourcing*

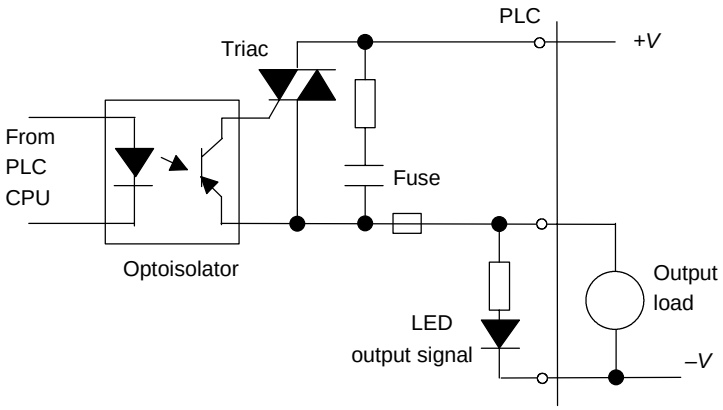


Figure 4.9 *Triac output unit*

Analogue outputs are frequently required and can be provided by digital-to-analogue converters (DACs) at the output channel. The input to the converter is a sequence of bits with each bit along a parallel line. Figure 4.10 shows the basic function of the converter.

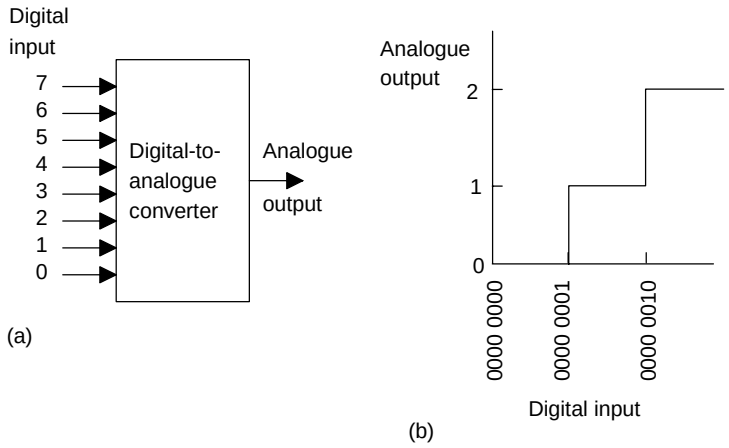


Figure 4.10 *(a) DAC function, (b) digital-to-analogue conversion*

A bit in the 0 line gives rise to a certain size output pulse. A bit in the 1 line gives rise to an output pulse of twice the size of the 0 line pulse. A bit in the 2 line gives rise to an output pulse of twice the size of the 1 line pulse. A bit in the 3 line gives rise to an output pulse of twice the size of the 2 line pulse, and so on. All the outputs add together to give the analogue version of the digital input. When the digital input changes, the analogue output changes in a stepped manner, the voltage changing by the voltage changes associated with each bit. For example, if we have an 8-bit converter then the output is made up of voltage values of $2^8 = 256$ analogue steps. Suppose the output range is set to 10 V d.c. One bit then gives a change of 10/255 V or about 0.04 V. Thus we have:

Digital input (V)	Analogue output (V)
00000000	0.00
00000001	0.04
00000010	0.08 + 0.00 = 0.08
00000011	0.08 + 0.04 = 0.12
00000100	0.16
00000101	0.016 + 0.00 + 0.04 = 0.20
00000110	0.016 + 0.08 = 0.24
00000111	0.016 + 0.08 + 0.04 = 0.28
00001000	0.32
etc.	

Analogue output modules are usually provided in a number of outputs, e.g. 4 to 20 mA, 0 to +5 V d.c., 0 to +10 V d.c., and the appropriate output is selected by switches on the module. Modules generally have outputs in two forms, one for which all the outputs from that module have a common voltage supply and one which drives outputs having their own individual voltage supplies. Figure 4.11 shows the basic principles of these two forms of output.

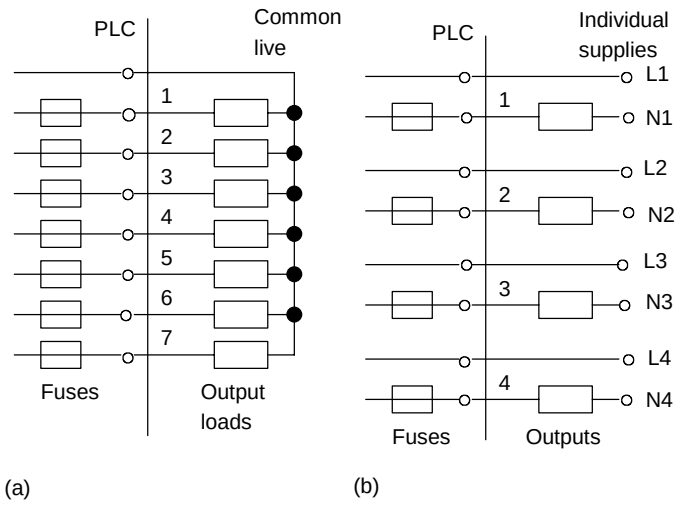


Figure 4.11 *Forms of output: (a) common supply, (b) individual supplies*

4.2 Signal conditioning

When connecting sensors which generate digital or discrete signals to an input unit, care has to be taken to ensure that voltage levels match. However, many sensors generate analogue signals. In order to avoid having a multiplicity of analogue input channels to cope with the wide diversity of analogue signals that can be generated by sensors, external signal conditioning is often used to bring analogue signals to a common range and so allow a standard form of analogue input channel to be used.

A common standard that is used (Figure 4.12) is to convert analogue signals to a current in the range 4 to 20 mA and thus to a voltage by passing it through a 250 Ω resistance to give a 1 to 5 V input signal. Thus, for example, a sensor used to monitor liquid level in the height range 0 to 1 m would have the 0 level represented by 4 mA and the 1 m represented by 20 mA. The use of 4 mA to represent the low end of the analogue range serves the purpose of distinguishing between when the sensor is indicating zero and when the sensor is not working and giving zero response for that reason. When this happens the current would be 0 mA. The 4 mA also is often a suitable current to operate a sensor and so eliminate the need for a separate power supply.

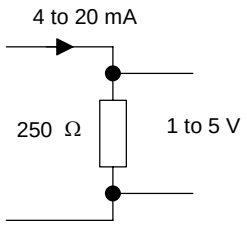


Figure 4.12 *Standard analogue signal*

A potential divider (Figure 4.13) can be used to reduce a voltage from a sensor to the required level; the output voltage level V_{out} is:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

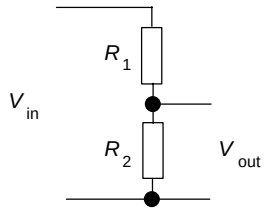


Figure 4.13 *Potential divider*

Amplifiers can be used to increase the voltage level; Figure 4.14 shows the basic form of the circuits that might be used with a 741 operational amplifier with (a) being an inverting amplifier and (b) a non-inverting amplifier. With the inverting amplifier the output V_{out} is:

$$V_{out} = -\frac{R_2}{R_1} V_{in}$$

and with the non-inverting amplifier:

$$V_{out} = \frac{R_1 + R_2}{R_1} V_{in}$$

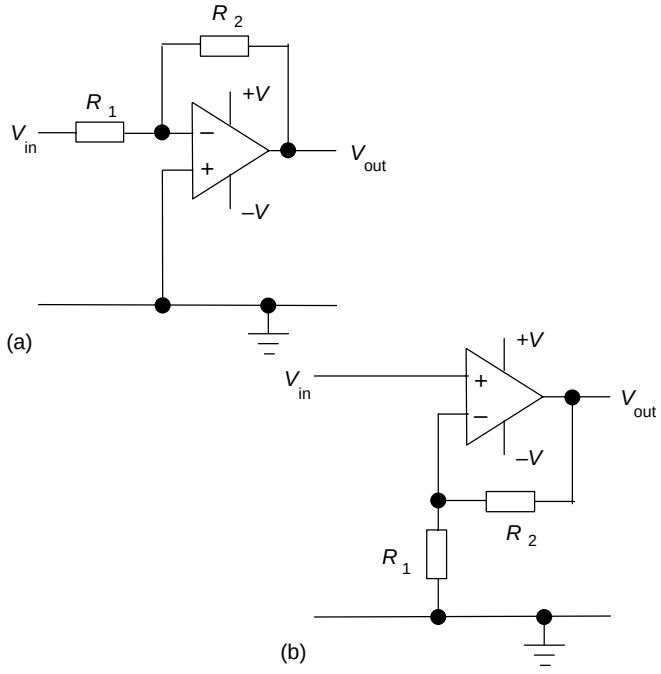


Figure 4.14 *Operational amplifier circuits*

Often a differential amplifier is needed to amplify the difference between two input voltages. Such is the case when a sensor, e.g. a strain gauge, is connected in a Wheatstone bridge and the output is the difference between two voltages or a thermocouple where the voltage difference between the hot and cold junctions is required. Figure 4.14 shows the basic form of an operational amplifier circuit for this purpose. The output voltage V_{out} is:

$$V_{out} = \frac{R_2}{R_1}(V_2 - V_1)$$

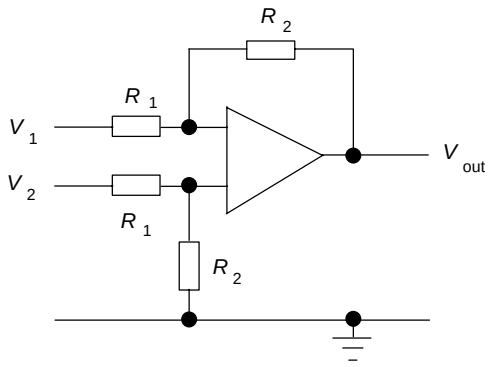


Figure 4.15 *Differential amplifier*

As an illustration of the use of signal conditioning, Figure 4.16 shows the arrangement that might be used for a strain gauge sensor. The sensor is connected in a Wheatstone bridge and the out-of-balance potential difference amplified by a differential amplifier before being fed via an analogue-to-digital converter unit which is part of the analogue input port of the PLC.

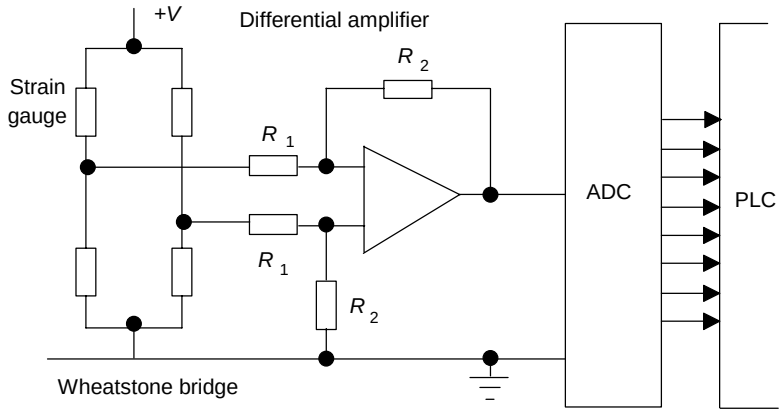


Figure 4.16 *Signal conditioning with a strain gauge sensor*

4.3 Remote connections

When there are many inputs or outputs located considerable distances away from the PLC, while it would be possible to run cables from each such device to the PLC a more economic solution is to use input/output modules in the vicinity of the inputs and outputs and use just a single core cable to connect each, over the long distances, to the PLC instead of the multicore cable that would be needed without such distant I/O modules (Figure 4.17).

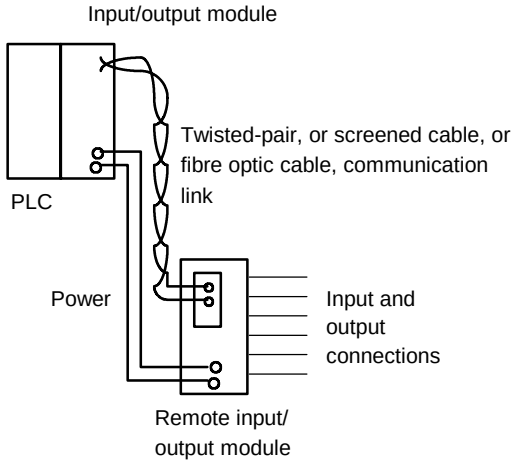


Figure 4.17 *Use of remote input/output module*

In some situations a number of PLCs may be linked together with a master PLC unit sending and receiving input/output data from the other units (Figure 4.18). The distant PLCs do not contain the control program since all the control processing is carried out by the master PLC.

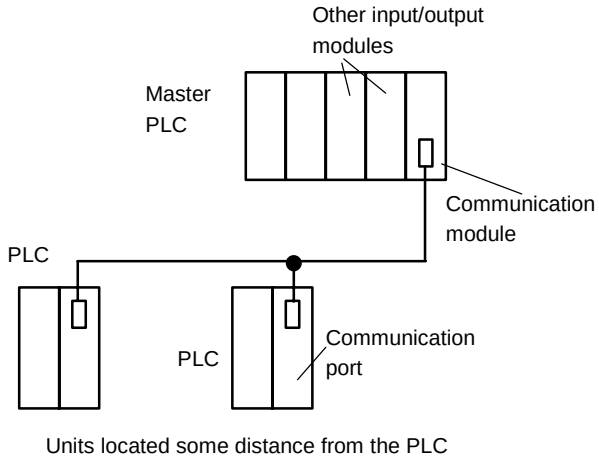


Figure 4.18 *Use of remote input/output PLC systems*

The cables used for communicating data between remote input/output modules and a central PLC, remote PLCs and the master PLC are

typically *twisted-pair cabling*, often routed through grounded steel conduit in order to reduce electrical ‘noise’. *Coaxial cable* enables higher data rates to be transmitted and does not require the shielding of steel conduit. *Fibre-optic cabling* has the advantage of resistance to noise, small size and flexibility and is now becoming more widely used.

4.3.1 Serial and parallel communications

Serial communication is when data is transmitted one bit at a time (Figure 4.19(a)). Thus if an 8-bit word is to be transmitted, the eight bits are transmitted one at a time in sequence along a cable. This means that a data word has to be separated into its constituent bits for transmission and then reassembled into the word when received. *Parallel communication* is when all the constituent bits of a word are simultaneously transmitted along parallel cables (Figure 4.19(b)). This allows data to be transmitted over short distances at high speeds.

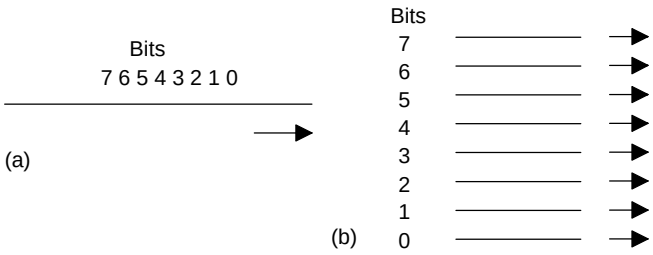


Figure 4.19 *(a) Serial communication, (b) parallel communication*

Serial communication is used for transmitting data over long distances. It is much cheaper to run, for serial communication, a single core cable over a long distance than the multicore cables that would be needed for parallel communication. With a PLC system, serial communication might be used for the connection between a computer, when used as a programming terminal, and a PLC. Parallel communication might be used when connecting laboratory instruments to the system. However, internally, PLCs work, for speed, with parallel communications. Thus, circuits called UARTS (universal asynchronous receivers–transmitters) have to be used at input/output ports to converts serial communications signals to parallel.

4.3.2 Serial standards

For successful serial communications to occur, it is necessary to specify:

1 The voltage levels to be used for signals, i.e. what signal represents a 0 and what represents a 1.
2 What the bit patterns being transmitted mean and how the message is built up. Bear in mind that a sequence of words are being sent along the same cable and it is necessary to be able to be able to determine when one word starts and finishes and the next word starts.
3 The speed at which the bit pattern is to be sent, i.e. the number of bits per second.

4 Synchronisation of the clocks at each end. This is necessary if, for example, a particular duration transmitted pulse it to be recognised by the receiver as just a single bit rather than two bits.
5 Protocols, or flow controls, to enable such information as ‘able to receive data’ or ‘not ready to receive data’ to be received. This is commonly done by using two extra signal wires (termed handshake wires), one to tell the receiver that the transmitter is ready to send the data and the other to tell the transmitter that the receiver is ready to receive data.
6 Error-checking to enable a bit pattern to be checked to determine if corruption of the data has occurred during transmission.

The most common standard serial communications interface used is the *RS232*. Connections are made via 25-pin D-type connectors (Figure 4.20) with usually, though not always, a male plug on cables and a female socket on the equipment. Not all the pins are used in every application.

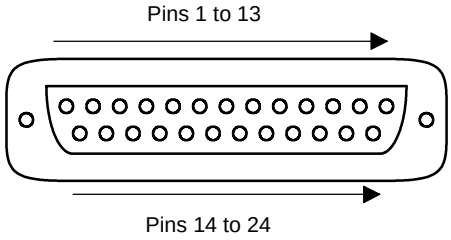


Figure 4.20 *D connector*

The minimum requirements are:

Pin 1: Ground connection to the frame of chassis
Pin 2: Serial transmitted data (output data pin)
Pin 3: Serial received data (input data pin)
Pin 7: Signal ground which acts as a common signal return path

A configuration that is widely used with interfaces involving computers is:

Pin 1: Ground connection to the frame of chassis
Pin 2: Serial transmitted data (output data pin)
Pin 3: Serial received data (input data pin)
Pin 4: Request to send
Pin 5: Clear to send
Pin 6: Data set ready
Pin 7: Signal ground which acts as a common signal return path
Pin 20: Data terminal ready

The signals sent through pins 4, 5, 6 and 20 are used to check that the receiving end is ready to receive a signal, the transmitting end is ready to send and the data is ready to be sent. With RS232, a 1 bit is represented by a voltage between –5 and –25 V, normally –12 V, and a 0 by a voltage between +5 and +25 V, normally +12 V.

The term *baud rate* is used to describe the transmission rate, it being approximately the number of bits transmitted or received per second. However, not all the bits transmitted can be used for data, some have to be used to indicate the start and stop of a serial piece of data, these often being termed *flags*, and as a check as to whether the data has been corrupted during transmission. Figure 4.21 shows the type of signal that might be sent with RS232. The parity bit is added to check whether corruption has occurred, with even parity a 1 being added to make the number of 1s an even number. To send seven bits of data, eleven bits may be required.

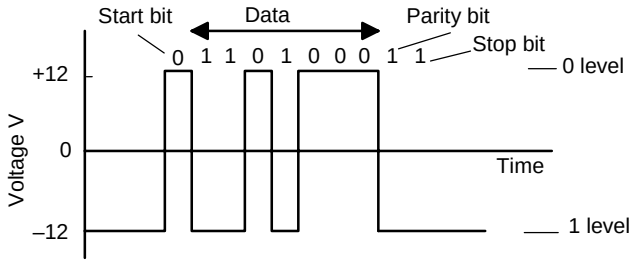


Figure 4.21 *RS232 signal levels*

Other standards such as the *RS422* and *RS423* are similar to RS232. RS232 is limited over the distances it can be used, noise limiting the transmission of high numbers of bits per second when the length of cable is more than about 15 m. RS422 can be used for longer distances. This uses a balanced method of transmission. Such circuits require two lines for the transmission, the transmitted signal being the voltage difference between the two lines. Noise affecting both lines equally will have no effect on the transmitted signal. Figure 4.22 shows how, for RS232 and RS422, the data rates that can be transmitted without noise becoming too significant depend on the distance. RS422 lines can be used for much greater distances than RS232.

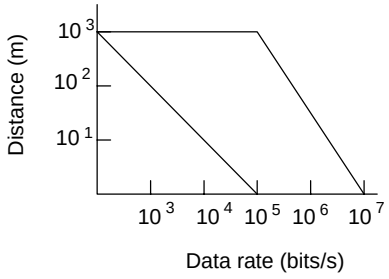


Figure 4.22 *Transmission with RS232 and RS422*

An alternative to RS422 is the *20 mA loop* which was an earlier standard and is still widely used for long distance serial communication, particularly in industrial systems where the communication path is likely to suffer from electrical noise (Figure 4.23). This system consists of a

circuit, a loop of wire, containing a current source. The serial data is transmitted by the current being switched on and off, a 0 being transmitted as zero current and a 1 as 20 mA.

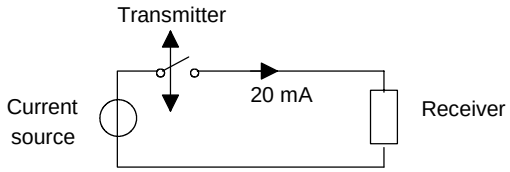


Figure 4.23 *20 mA loop*

4.3.3 Parallel standards

The standard interface most commonly used for parallel communications is *IEEE-488*. This was originally developed by Hewlett Packard to link their computers and instruments and was known as the *Hewlett Packard Instrumentation Bus*. It is now often termed the *General Purpose Instrument Bus*. This bus provides a means of making interconnections so that parallel data communications can take place between listeners, talkers and controllers. Listeners are devices that accept data from the bus, talkers place data, on request, on the bus and controllers manage the flow of data on the bus and provide processing facilities. There is a total of 24 lines, of which eight bi-directional lines are used to carry data and commands between the various devices connected to the bus, five lines are used for control and status signals, three are used for handshaking between devices and eight are ground return lines (Figure 4.24).

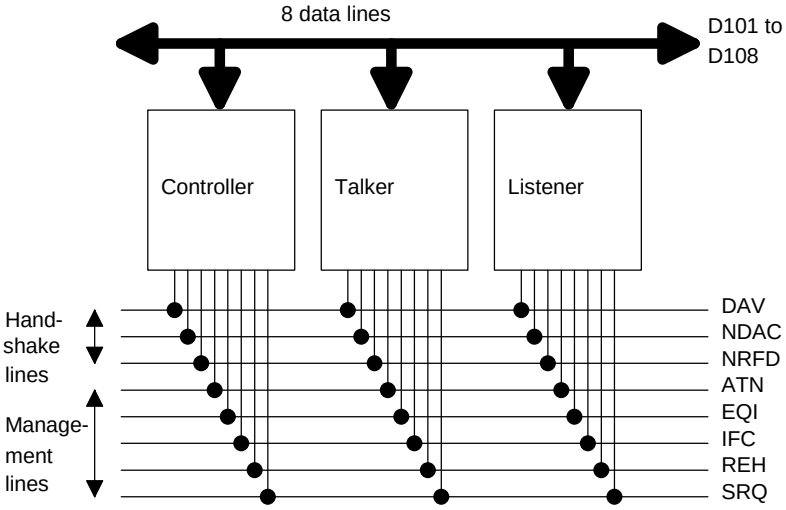


Figure 4.24 *The IEEE-488 bus structure*

Commands from the controller are signalled by taking the Attention Line (ATN) low, otherwise it is high, and thus indicating that the data lines contain data. The commands can be directed to individual devices by placing addresses on the data lines. Each device on the bus has its own

address. Device addresses are sent via the data lines as a parallel 7-bit word, the lowest 5-bits providing the device address and the other two bits control information. If both these bits are 0 then the commands are sent to all addresses, if bit 6 is 1 and bit 7 a 0 the addressed device is switched to be a listener, if bit 6 is 0 and bit 7 is 1 then the device is switched to be a talker.

As illustrated above by the function of the ATN line, the management lines each have an individual task in the control of information. The handshake lines are used for controlling the transfer of data. The three lines ensure that the talker will only talk when it is being listened to by listeners. Table 4.1 lists the functions of all the lines and their pin numbers in a 25-way D-type connector.

Table 4.1 *IEEE-488 bus system*

Pin	Signal group	Abbreviation	Signal/function
1	Data	D101	Data line 1.
2	Data	D102	Data line 2.
3	Data	D103	Data line 3.
4	Data	D104	Data line 4.
5	Management	EOI	End Or Identify. This is used to either signify the end of a message sequence from a talker device or is used by the controller to ask a device to identify itself.
6	Handshake	DAV	Data valid. When the level is low on this line then the information on the data bus is valid and acceptable.
7	Handshake	NRFD	Not Ready For Data. This line is used by listener devices taking it high to indicate that they are ready to accept data.
8	Handshake	NDAC	Not Data Accepted. This line is used by listeners taking it high to indicate that data is being accepted.
9	Management	IFC	Interface Clear. This is used by the controller to reset all the devices of the system to the start state.
10	Management	SRQ	Service Request. This is used by devices to signal to the controller that they need attention.
11	Management	ATN	Attention. This is used by the controller to signal that it is placing a command on the data lines.
12		SHIELD	Shield.
13	Data	D105	Data line 5.
14	Data	D106	Data line 6.
15	Data	D107	Data line 7.
16	Data	D108	Data line 8.
17	Management	REN	Remote Enable. This enables a device on the bus to indicate that it is to be selected for remote control rather than by its own control panel.
18		GND	Ground/common.
19		GND	Ground/common.
20		GND	Ground/common.
21		GND	Ground/common.
22		GND	Ground/common.
23		GND	Ground/common.
24		GND	Ground/common.

Figure 4.25 shows the handshaking sequence that occurs when data is put on the data lines. Initially DAV is high, indicating that there is no valid data on the data bus, NRFD and NDAC also being low. When a data word is put on the data lines, NRFD is made high to indicate that all listeners are ready to accept data and DAV is made low to indicate that new data is on the data lines. When a device accepts a data word it sets NDAC high to indicate that is has accepted the data and NRFD low to indicate that it is now not ready to accept data. When all the listeners have set NDAC high, then the talker cancels the data valid signal, DAV going high. This then results in NDAC being set low. The entire process can then be repeated for another word being put on the data bus.

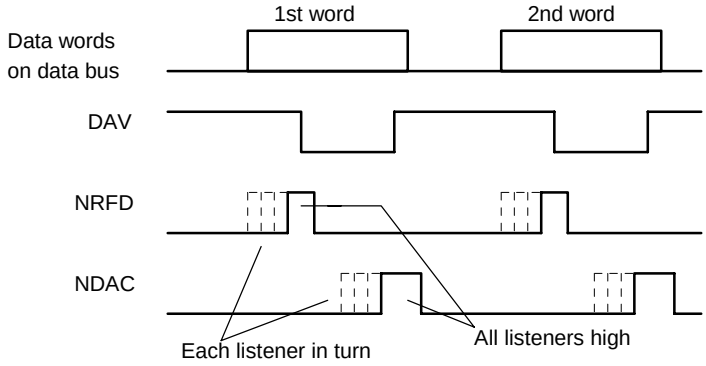


Figure 4.25 *Handshaking sequence*

4.3.4 Protocols

It is necessary to exercise control of the flow of data between two devices so what constitutes the message, and how the communication is to be initiated and terminated, is defined. This is termed the *protocol*.

Thus one device needs to indicate to the other to start or stop sending data. This can be done by using handshaking wires connecting transmitting and receiving devices so that a signal along one such wire can tell the receiver that the transmitter is ready to send (RTS) and along another wire that the transmitter is ready to receive, a clear to send signal (CTS). RTS and CTS lines are provided for in RS232 serial communication links.

An alternative is to use additional characters on the transmitting wires. With the ENQ/ACK protocol, data packets are sent to a receiver with a query character ENQ. When this character is received the end of the data packet has been reached. Once the receiver has processed that data, it can indicate it is ready for another block of data by sending back an acknowledge (ACK) signal. Another form, the XON/XOFF, has the receiving device sending a XOFF signal to the sending device when it wishes the data flow to cease. The transmitter then waits for an XON signal before resuming transmission.

One form of checking for errors in the message that might occur as a result of transmission is the *parity check*. This is an extra bit added to a message to ensure that the number of bits in a piece of data is always odd

or always even. For example, 0100100 is even since there is an even number of 1s and 0110100 is odd since there is an odd number of 1s. To make both these odd parity then the extra bit added at the end in the first case is 1 and in the second case 0, i.e. we have 01001001 and 01101000. Thus when a message is sent out with odd bit parity, if the receiver finds that the bits give an even sum, then the message has been corrupted during transmission and the receiver can request that the message be repeated.

The parity bit method can detect if there is an error resulting from a single 0 changing to a 1 or a 1 changing to a 0 but cannot detect two such errors occurring since there is then no change in parity. To check on such occurrences more elaborate checking methods have to be used. One method involves storing data words in an array of rows and columns. Parity can then be checked for each row and each column. The following illustrates this for seven words using even parity.

		Row parity bits
Column parity bits	00101010	1
↑	10010101	0
	10100000	0
Block	01100011	0
of data	11010101	1
	10010101	1
↓	00111100	0

Another method, termed *cyclic redundancy check codes*, involves splitting the message into blocks. Each block is then treated as a binary number and is divided by a predetermined number. The remainder from this division is then sent as the error checking number on the conclusion of the message and enables a check on the accuracy of the message to be undertaken.

4.3.5 ASCII codes

The most widely used code for the transmission of characters is the ASCII code (American Standard Code for Information Interchange). This is a seven-bit code giving 128 different combinations of bits covering lower case and upper case alphanumeric characters, punctuation and 32 control codes. As an illustration, Table 4.2 shows the codes used for capital letters. Examples of control codes are SOH, for start of heading, i.e. the first character of a heading of an information message, as 000 0001; STX, for start of text, as 000 0010; ETX, for end of text, as 000 0011; EOT, for end of transmission, as 000 0011.

4.4 Networks

The increasing use of automation in industry has led to the need for communications and control on a plant-wide basis with programmable controllers, computers, robots, and CNC machines interconnected. The term *local area network (LAN)* is used to describe a communications network designed to link computers and their peripherals within the same building or site.

Table 4.2 *Examples of ASCII codes*

	ASCII		ASCII		ASCII
A	100 0001	N	100 1110	0	011 0000
B	100 0010	O	100 1111	1	011 0001
C	100 0011	P	101 0000	2	011 0010
D	100 0100	Q	101 0001	3	011 0011
E	100 0101	R	101 0010	4	011 0100
F	100 0110	S	101 0011	5	011 0101
G	100 0111	T	101 0100	6	011 0110
H	100 1000	U	101 0101	7	011 0111
I	100 1001	V	101 0110	8	011 1000
J	100 1010	W	101 0111	9	011 1001
K	100 1011	X	101 1000		
L	100 1100	Y	101 1001		
M	100 1101	Z	101 1010		

Networks can take three basic forms. With the *star* form (Figure 4.26(a)) the terminals are each directly linked to a central computer, termed the host, or master with the terminals being termed slaves. The host contains the memory, processing and switching equipment to enable the terminals to communicate. Access to the terminals is by the host asking each terminal in turn whether it wants to talk or listen. With the *bus* or *single highway* type of network (Figure 4.26(b)), each of the terminals is linked into a single cable and so each transmitter/receiver has a direct path to each other transmitter/receiver in the network. Methods, i.e. protocols, have to be adopted to ensure that no more than one terminal talks at once, otherwise confusion can occur. A terminal has to be able to detect whether another terminal is talking before it starts to talk. With the *ring* network (Figure 4.26(c)), a continuous cable, in the form of a ring, links all the terminals. Again methods have to be employed to enable communications from different terminals without messages becoming mixed up. The single highway and the ring methods are often termed *peer to peer* in that each terminal has equal status. Such a system allows many stations to use the same network.

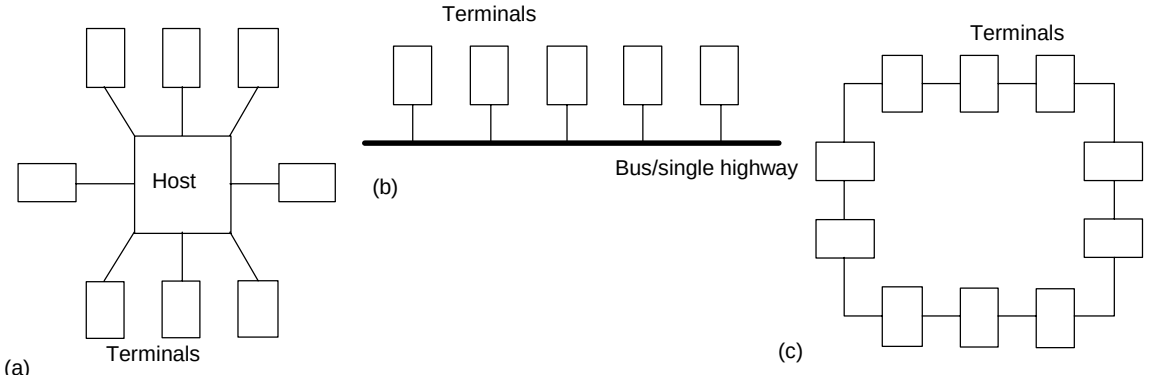


Figure 4.26 *Networks: (a) star, (b) bus/single highway, (c) ring*

With ring-based networks, two commonly used methods that are employed to avoid two stations talking at once and so giving rise to confusion are token passing and slot passing. With token passing, a special bit pattern called a token is circulated round the network. When a station wishes to transmit into the network it waits until it receives the token, then transmits the data with the token attached. Another station wishing to transmit cannot do so until the token has been freed by removal from the data by a receiver. With slot passing, empty slots are circulated into which stations can deposit data for transmission.

Bus systems generally employ the method in which a system wishing to transmit listens to see if any messages are being transmitted. If no message is being transmitted, a station can take control of the network and transmit its message. This method is known as *carrier sense multiple access* (CSMA). However, we could end up with two stations simultaneously perceiving the network to be clear for transmission and both simultaneously taking control and sending messages. The result would be a 'collision' of their transmitted data and corruption. If such a situation is detected, both stations cease transmitting and wait a random time before attempting to again transmit. This is known as *carrier sense multiple access with collision detection* (CSMA/CD).

Different PLC manufacturers adopt different forms of network systems and methods of communication for use with their PLCs. For example, Mitsubishi uses a network termed MelsecNET, Allen Bradley uses Data Highway Plus, General Electric uses GENET, Texas Instruments uses TIWAY and Siemens has four forms under the general name SINEC. Most employ peer to peer forms, e.g. Allen-Bradley. Siemens has two low level forms, SINECL1 which is a star, i.e. master-slave form, and SINECL2 which is peer to peer.

4.4.1 Distributed systems

Often PLCs figure in an entire hierarchy of communications (Figure 4.27). Thus at the lowest level we have input and output devices such as sensor and motors interfaced through I/O interfaces with the next level. The next level involves controllers such as small PLCs or small computers, linked through a network with the next level of larger PLCs and computers exercising local area control. These in turn may be part of a network involved with a large mainframe company computer controlling all.

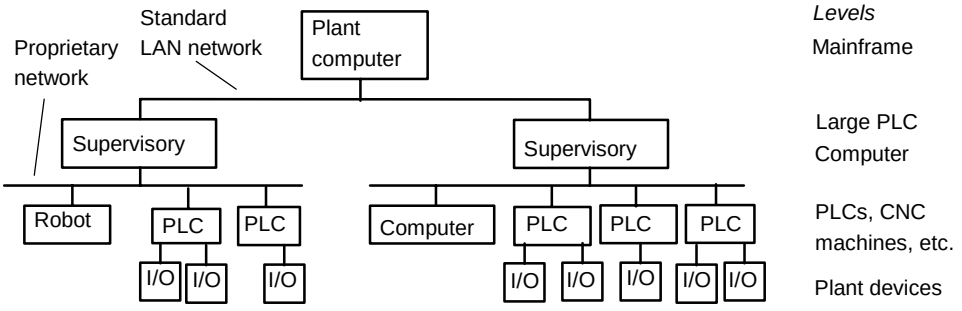


Figure 4.27 *Control hierarchy*

There is increasing use made of systems that can both control and monitor industrial processes. This involves control and the gathering of data. The term SCADA, which stands for *supervisory control and data acquisition system*, is widely used for such a system.

4.4.2 Network standards

Interconnecting several devices can present problems because of compatibility problems, e.g. they may operate at different baud rates or use different protocols. In order to facilitate communications between different devices the International Standards Organisation (ISO) in 1979 devised a model to be used for standardisation for open systems interconnection (OSI); the model is termed the *ISO/OSI model*. A communication link between items of digital equipment is defined in terms of physical, electrical, protocol and user standards, the ISO/OSI model breaking this down into seven layers (Figure 4.28).

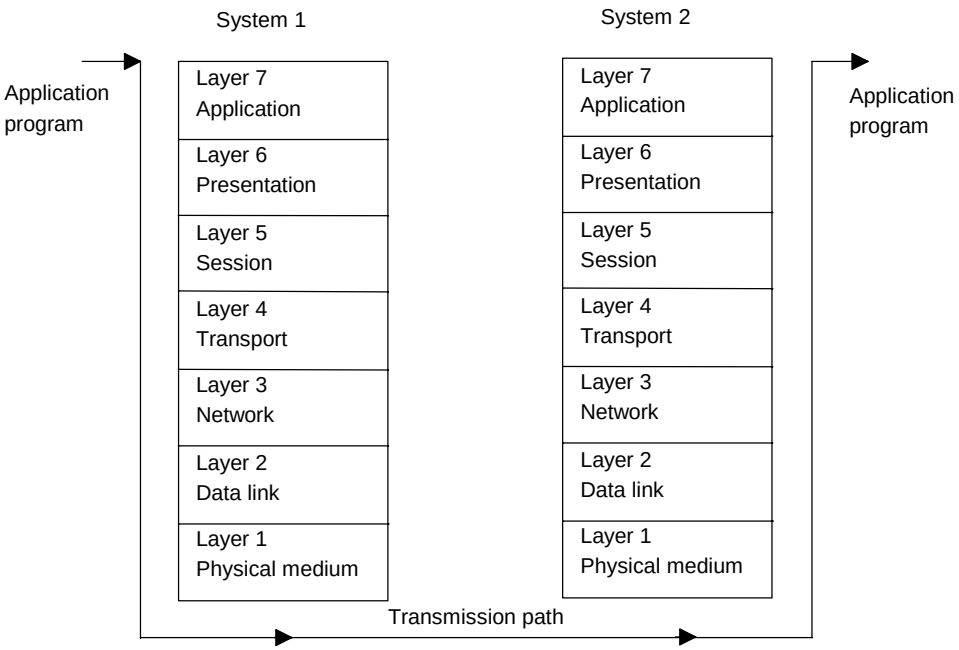


Figure 4.28 *ISO/OSI model*

The function of each layer in the model is:

Layer 1: Physical medium
This layer is concerned with the coding and physical transmission of information. Its functions include synchronising data transfer and transferring bits of data between systems.

Layer 2: Data link
This layer defines the protocols for sending and receiving information between systems that are directly connected to each other. Its functions include assembling bits from the physical layer into blocks

and transferring them, controlling the sequence of data blocks and detecting and correcting errors.

Layer 3: Network
This layer defines the switching that routes data between systems in the network.

Layer 4: Transport
This layer defines the protocols responsible for sending messages from one end of the network to the other. It controls message flow.

Layer 5: Session
This layer provides the function to set up communications between users at separate locations.

Layer 6: Presentation
This layer assures that information is delivered in an understandable form.

Layer 7: Application
This layer has the function of linking the user program into the communication process and is concerned with the meaning of the transmitted information.

Each layer is self-contained and only deals with the interfaces of the layer immediately above and below it; it performs its tasks and transfers its results to the layer above or the layer below. It thus enables manufacturers of products to design products operable in a particular layer that will interface with the hardware of other manufacturers.

In 1980, the IEEE (Institute of Electronic and Electrical Engineers) began Project 802. This is a model which adheres to the OSI Physical layer but subdivided the Data link layer into two separate layers: the Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC layer defines the access method to the transmission medium and consists of a number of standards to control access to the network and ensure that only one user is able to transmit at any one time. One standard is IEEE 802.3 Carrier Sense Multiple Access and Collision Detection (CSMA/CD); stations have to listen for other transmissions before being able to gain control of the network and transmit. Another standard is IEEE 802.4 Token Passing Bus; with this method a special bit pattern, the token, is circulated and when a station wishes to transmit it waits until it receives the token and then attaches it to the end of the data. The LLC layer is responsible for the reliable transmission of data packets across the Physical layer.

4.4.3 Examples of commercial systems

General Motors in the United States had a problem in automating their manufacturing activities by 1990 of requiring all their systems to be able to talk to each other. They thus developed a standard communications system for factory automation applications, this being termed the *manufacturing automation protocol* (MAP). The system was for all systems on the shop floor, e.g. robot systems, PLCs, welding systems. Table 4.2 shows the MAP model and its relationship to the ISO model. In order for non-OSI equipment to operate on the MAP system, gateways may be used. These are self-contained units or interface boards that fit in

the device so that messages from a non-OSI network/device may be transmitted through the MAP broad band token bus to other systems.

Table 4.3 *MAP*

ISO layer		MAP protocol
7	Application	ISO file transfer, MMFS, FTAM, CASE
6	Presentation	
5	Session	ISO session kernel
4	Transport	ISO transport class 4
3	Network	ISO Internet
2	Data link	IEEE 802.2 class 1; IEEE 802.4 token bus
1	Physical	IEEE 802.4 broad band
	Transmission	10 mbps coaxial cable with RF modulators

Note: MMFS = manufacturing message format standard, FTAM = file transfer, CASE = common applications service; each of these provides a set of commands that will be understood by devices and the software used.

For the data link, methods are needed to ensure that only the user of the network is able to transmit at any one time and for MAP the method used is token passing. The term *broad band* is used for a network in which information is modulated onto a radio frequency carrier which is then transmitted through the coaxial cable.

MAP is not widely used, a more commonly used system being the *Ethernet*. This is a single bus system with CSMA/CD used to control access. It uses coaxial cable with a maximum length of 500 m and up to 1024 stations can be accommodated, repeaters which restore signal amplitude, waveform and timing can be used to extend this capability (Figure 4.29). Each station is connected to the bus via a transceiver, the transceiver clamping on to the bus cable. The term ‘vampire tap’ is used for the clamp on to the cable since stations can be connected or removed without disrupting system operation.

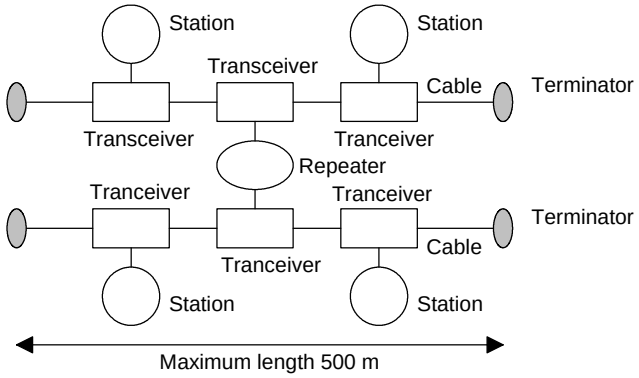


Figure 4.29 *Baseband Ethernet with repeaters. The term base band is used when the signal is transmitted as just a series of voltage levels directly representing the bits being transmitted.*

Ethernet does not have a master station, each connected station being of equal status and so we have peer-to-peer communication. A station wishing to send a message on the bus will determine whether the bus is clear and, when it is, put its message frame on the bus. There is the slight probability that more than one station will sense an idle bus and attempt to transmit. Thus each sender monitors the bus during transmission and detects when the signal on the bus does not match its own output. When such a 'collision' is detected, the transmission continues for a short while in order to give time to other stations to detect the collision and then attempts to retransmit at a later time. Each message includes a bit sequence to indicate the destination address, source address, the data to be transmitted and a message check sequence. The message check sequence contains the cycle redundancy check (see Section 4.3.4). At each receiving station the frame's destination address is checked to determine whether the frame is destined for it. If so, it accepts the message. Ethernet is widely used where systems involve PLCs having to communicate with computers. The modular Allen-Bradley PLC-5 can be configured for use with a range of communication networks by the addition of suitable modules (see Figure 1.15), a module enabling use with Ethernet.

PLC manufacturers often have their own networks, in addition to generally offering the possibility of Ethernet. The *Allen-Bradley data highway* is a peer-to-peer system developed for Allen-Bradley PLCs and uses token passing to control message transmission. The station addresses of each PLC are set by switches on each PLC. Communication is established by a single message on the data highway, this specifying the sending and receiving addresses and the length of block to be transferred.

4.5 Processing inputs

A PLC is continuously running through its program and updating it as a result of the input signals. Each such loop is termed a *cycle*. PLCs could be operated by each input being examined as it occurred in the program and its effect on the program determined and the output correspondingly changed. This mode of operation is termed *continuous updating*.

Because, with continuous updating, there is time spent interrogating each input in turn, the time taken to examine several hundred input/output points can become comparatively long. To allow a more rapid execution of a program, a specific area of RAM is used as a buffer store between the control logic and the input/output unit. Each input/output has an address in this memory. At the start of each program cycle the CPU scans all the inputs and copies their status into the input/output addresses in RAM. As the program is executed the stored input data is read, as required, from RAM and the logic operations carried out. The resulting output signals are stored in the reserved input/output section of RAM. At the end of each program cycle all the outputs are transferred from RAM to the appropriate output channels. The outputs then retain their status until the next updating. This method of operation is termed *mass I/O copying*. The sequence can be summarised as (Figure 4.30):

1 Scan all the inputs and copy into RAM
2 Fetch and decode and execute all program instructions in sequence, copying output instructions to RAM

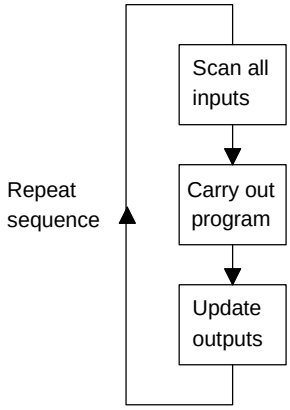


Figure 4.30 *PLC operation*

3 Update all outputs
4 Repeat the sequence

The time taken to complete a cycle of scanning inputs and updating outputs according to the program instructions, i.e. the *cycle time*, though relatively quick, is not instantaneous and means that the inputs are not watched all the time but samples of their states taken periodically. A typical cycle time is of the order of 10 to 50 ms. This means that the inputs and outputs are updated every 10 to 50 ms and thus there can be a delay of this order in the system reacting. It also means that if a very brief input cycle appears at the wrong moment in the cycle, it could be missed. In general, any input must be present for longer than the cycle time. Special modules are available for use in such circumstances.

Consider a PLC with a cycle time of 40 ms. What is the maximum frequency of digital impulses that can be detected? The maximum frequency will be if one pulse occurs every 40 ms, i.e. a frequency of 1/0.04 = 25 Hz.

The cycle or scanning time for a PLC, i.e. its response speed, is determined by:

1 The CPU used.
2 The size of the program to be scanned.
3 The number of input/outputs to be read.
4 The system functions that are in use, the greater the number the slower the scanning time.

As an illustration, the Mitsubishi compact PLC, MELSEC FX3U (see Section 1.4), has a quoted program cycle time of 0.065 μs per logical instruction. Thus the more complex the program the longer the cycle time will be.

4.6 I/O addresses

The PLC has to be able to identify each particular input and output. It does this by allocating addresses to each input and output. With a small

PLC this is likely to be just a number, prefixed by a letter to indicate whether it is an input or an output. Thus for the Mitsubishi PLC we might have inputs with addresses X400, X401, X402, etc., and outputs with addresses Y430, Y431, Y432, etc., the X indicating an input and the Y an output. Toshiba use a similar system.

With larger PLCs having several racks of input and output channels, the racks are numbered. With the Allen-Bradley PLC-5, the rack containing the processor is given the number 0 and the addresses of the other racks are numbered 1, 2, 3, etc. according to how set-up switches are set. Each rack can have a number of modules and each one deals with a number of inputs and/or outputs. Thus addresses can be of the form shown in Figure 4.31. For example, we might have an input with address I:012/03. This would indicate an input, rack 01, module 2 and terminal 03.

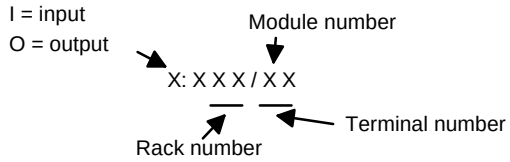


Figure 4.31 *Allen-Bradley PLC-5 addressing*

With the Siemens SIMATIC S5, the inputs and outputs are arranged in groups of 8. Each 8 group is termed a byte and each input or output within an 8 is termed a bit. The inputs and outputs thus have their addresses in terms of the byte and bit numbers, effectively giving a module number followed by a terminal number, a full stop (.) separating the two numbers. Figure 4.32 shows the system. Thus I0.1 is an input at bit 1 in byte 0, Q2.0 is an output at bit 0 in byte 2.

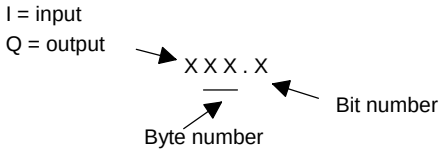


Figure 4.32 *Siemens SIMATIC S5 addressing*

The GEM-80 PLC assigns inputs and output addresses in terms of the module number and terminal number within that module. The letter A is used to designate inputs and B outputs. Thus A3.02 is an input at terminal 02 in module 3, B5.12 is an output at terminal 12 in module 5.

In addition to using addresses to identify inputs and outputs, PLCs also use their addressing systems to identify internal, software-created devices, such as relays, timers and counters.

Problems

Questions 1 to 9 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

1 Decide whether each of these statements is True (T) or False (F).

A serial communication interface:
(i) Involves data being transmitted and received one bit at a time.
(ii) Is a faster form of transmission than parallel communication.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

2 Decide whether each of these statements is True (T) or False (F).

The RS232 communications interface:
(i) Is a serial interface.
(ii) Is typically used for distances up to about 15 m.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 3 and 4 refer to the following which shows the bits on an RS232 data line being used to transmit the data 1100001:

0110000111
X YZ

3 Decide whether each of these statements is True (T) or False (F).

The extra bits X and Z at the beginning and the end are:
(i) To check whether the message is corrupted during transmission.
(ii) To indicate where the data starts and stops.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

4 Decide whether each of these statements is True (T) or False (F).

Bit Y is:
(i) The parity bit showing odd parity.
(ii) The parity bit showing even parity.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

5 Decide whether each of these statements is True (T) or False (F).

The parallel data communication interface:
(i) Enables data to be transmitted over short distances at high speeds.
(ii) A common standard is IEEE-488.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

6 Decide whether each of these statements is True (T) or False (F).

For communications over distances of the order of 100 to 300 m with a high transmission rate:
(i) The RS232 interface can be used.
(ii) The 20 mA current loop can be used.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

7 Decide whether each of these statements is True (T) or False (F).

With input/output processing, mass input/output copying:
(i) Scans all the inputs and copies their states into RAM.
(ii) Is a faster process than continuous updating.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

8 The cycle time of a PLC is the time it takes to:

A Read an input signal.
B Read all the input signals.
C Check all the input signals against the program.
D Read all the inputs, run the program and update all outputs.

9 Decide whether each of these statements is True (T) or False (F).

A PLC with a long cycle time is suitable for:
(i) Short duration inputs.
(ii) High frequency inputs.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

10 Specify (a) the odd parity bit, (b) the even parity bit, to be used when the data 1010100 is transmitted.

11 Explain the purpose of using a parity bit.

12 Explain the continuous updating and the mass input/output copying methods of processing inputs/outputs.

5 Ladder and functional block programming

Programs for microprocessor-based systems have to be loaded into them in *machine code*, this being a sequence of binary code numbers to represent the program instructions. However, *assembly language* based on the use of mnemonics can be used, e.g. LD is used to indicate the operation required to load the data that follows the LD, and a computer program called an assembler is used to translate the mnemonics into machine code. Programming can be made even easier by the use of the so-called *high level languages*, e.g. C, BASIC, PASCAL, FORTRAN, COBOL. These use pre-packaged functions, represented by simple words or symbols descriptive of the function concerned. For example, with C language the symbol & is used for the logic AND operation. However, the use of these methods to write programs requires some skill in programming and PLCs are intended to be used by engineers without any great knowledge of programming. As a consequence, *ladder programming* was developed. This is a means of writing programs which can then be converted into machine code by some software for use by the PLC microprocessor.

This method of writing programs became adopted by most PLC manufacturers, however each tended to have developed their own versions and so an international standard has been adopted for ladder programming and indeed all the methods used for programming PLCs. The standard, published in 1993, is IEC 1131-3 (International Electrotechnical Commission). The IEC 1131-3 programming languages are ladder diagrams (LAD), instruction list (IL), sequential function charts (SFC), structured text (ST), and function block diagrams (FBD).

This chapter is an introduction to the programming of a PLC using ladder diagrams and functional block diagrams, with discussion of the other techniques in the next chapter. Here we are concerned with the basic techniques involved in developing ladder and function block programs to represent basic switching operations, involving the logic functions of AND, OR, Exclusive OR, NAND and NOR, and latching. Later chapters continue with further ladder programming involving other elements.

5.1 Ladder diagrams

As an introduction to ladder diagrams, consider the simple wiring diagram for an electrical circuit in Figure 5.1(a). The diagram shows the circuit for switching on or off an electric motor. We can redraw this diagram in a different way, using two vertical lines to represent the input power rails and stringing the rest of the circuit between them. Figure 5.1(b) shows the result. Both circuits have the switch in series with the motor and supplied

with electrical power when the switch is closed. The circuit shown in Figure 5.1(b) is termed a *ladder diagram*.

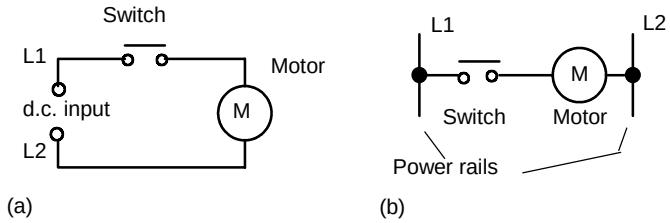


Figure 5.1 *Ways of drawing the same electrical circuit*

With such a diagram the power supply for the circuits is always shown as two vertical lines with the rest of the circuit as horizontal lines. The power lines, or rails as they are often termed, are like the vertical sides of a ladder with the horizontal circuit lines like the rungs of the ladder. The horizontal rungs show only the control portion of the circuit, in the case of Figure 5.1 it is just the switch in series with the motor. Circuit diagrams often show the relative physical location of the circuit components and how they are actually wired. With ladder diagrams no attempt is made to show the actual physical locations and the emphasis is on clearly showing how the control is exercised.

Figure 5.2 shows an example of a ladder diagram for a circuit that is used to start and stop a motor using push buttons. In the normal state, push button 1 is open and push button 2 closed. When button 1 is pressed, the motor circuit is completed and the motor starts. Also, the holding contacts wired in parallel with the motor close and remain closed as long as the motor is running. Thus when the push button 1 is released, the holding contacts maintain the circuit and hence the power to the motor. To stop the motor, button 2 is pressed. This disconnects the power to the motor and the holding contacts open. Thus when push button 2 is released, there is still no power to the motor. Thus we have a motor which is started by pressing button 1 and stopped by pressing button 2.

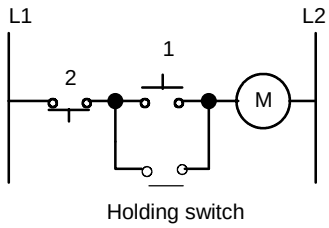


Figure 5.2 *Stop-start switch*

5.1.1 PLC ladder programming

A very commonly used method of programming PLCs is based on the use of *ladder diagrams*. Writing a program is then equivalent to drawing a switching circuit. The ladder diagram consists of two vertical lines representing the power rails. Circuits are connected as horizontal lines, i.e. the rungs of the ladder, between these two verticals.

In drawing a ladder diagram, certain conventions are adopted:

1 The vertical lines of the diagram represent the power rails between which circuits are connected. The power flow is taken to be from the left-hand vertical across a rung.
2 Each rung on the ladder defines one operation in the control process.
3 A ladder diagram is read from left to right and from top to bottom, Figure 5.3 showing the scanning motion employed by the PLC. The top rung is read from left to right. Then the second rung down is read from left to right and so on. When the PLC is in its run mode, it goes through the entire ladder program to the end, the end rung of the program being clearly denoted, and then promptly resumes at the start (see Section 4.4). This procedure of going through all the rungs of the program is termed a *cycle*. The end rung might be indicated by a block with the word END or RET for return, since the program promptly returns to its beginning.

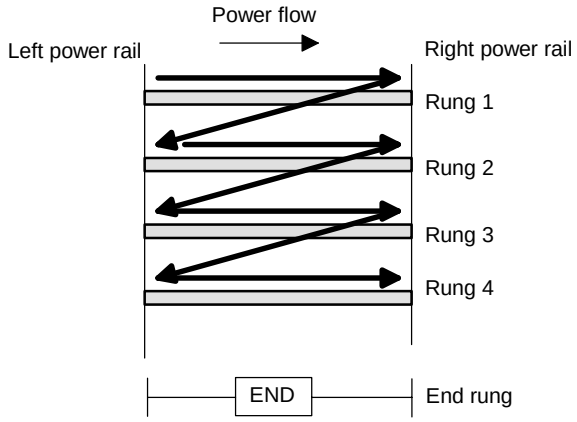


Figure 5.3 *Scanning the ladder program*

4 Each rung must start with an input or inputs and must end with at least one output. The term input is used for a control action, such as closing the contacts of a switch, used as an input to the PLC. The term output is used for a device connected to the output of a PLC, e.g. a motor.
5 Electrical devices are shown in their normal condition. Thus a switch which is normally open until some object closes it, is shown as open on the ladder diagram. A switch that is normally closed is shown closed.
6 A particular device can appear in more than one rung of a ladder. For example, we might have a relay which switches on one or more devices. The same letters and/or numbers are used to label the device in each situation.
7 The inputs and outputs are all identified by their addresses, the notation used depending on the PLC manufacturer. This is the address of the input or output in the memory of the PLC (see Section 4.6).

Figure 5.4 shows standard IEC 1131-3 symbols that are used for input and output devices. Some slight variations occur between the symbols when used in semi-graphic form and when in full graphic. Note that inputs are represented by different symbols representing normally open or normally closed contacts. The action of the input is equivalent to opening or closing a switch. Output coils are represented by just one form of symbol. Further symbols will be introduced in later chapters.

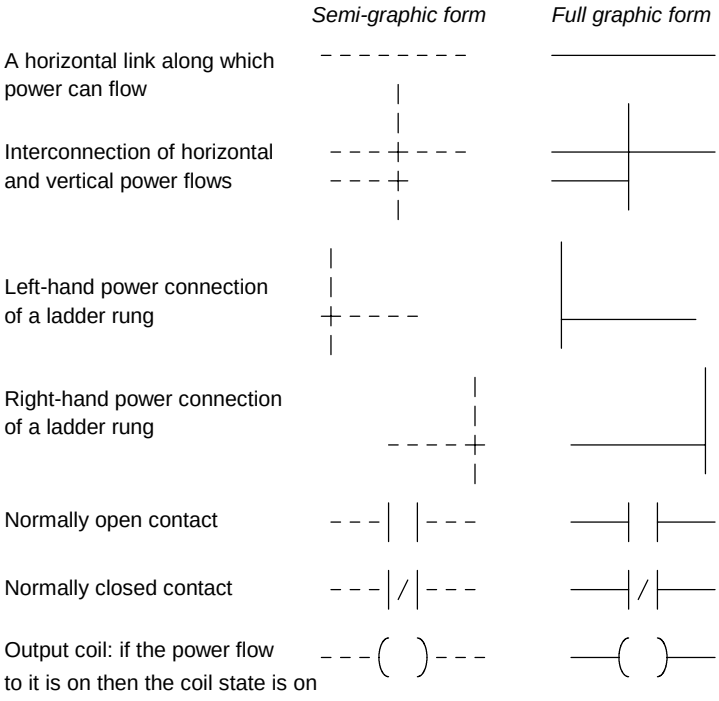

Semi-graphic form
Full graphic form
A horizontal link along which power can flow
Interconnection of horizontal and vertical power flows
Left-hand power connection of a ladder rung
Right-hand power connection of a ladder rung
Normally open contact
Normally closed contact
Output coil: if the power flow to it is on then the coil state is on

Figure 5.4 *Basic symbols*

To illustrate the drawing of the rung of a ladder diagram, consider a situation where the energising of an output device, e.g. a motor, depends on a normally open start switch being activated by being closed. The input is thus the switch and the output the motor. Figure 5.5(a) shows the ladder diagram.

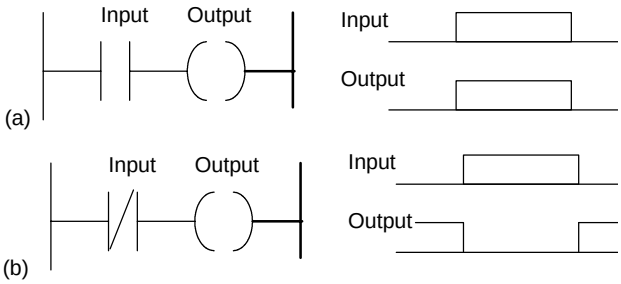


Figure 5.5 *A ladder rung*

Starting with the input, we have the normally open symbol | | for the input contacts. There are no other input devices and the line terminates with the output, denoted by the symbol ( ). When the switch is closed, i.e. there is an input, the output of the motor is activated. Only while there is an input to the contacts is there an output. If there had been a normally closed switch |/| with the output (Figure 5.5(b)), then there would have been an output until that switch was opened. Only while there is no input to the contacts is there an output.

In drawing ladder diagrams the names of the associated variable or addresses of each element are appended to its symbol. Thus Figure 5.6 shows how the ladder diagram of Figure 5.5(a) would appear using (a) Mitsubishi, (b) Siemens, (c) Allen-Bradley, (d) Telemecanique notations for the addresses. Thus Figure 5.6(a) indicates that this rung of the ladder program has an input from address X400 and an output to address Y430. When wiring up the inputs and outputs to the PLC, the relevant ones must be connected to the input and output terminals with these addresses.

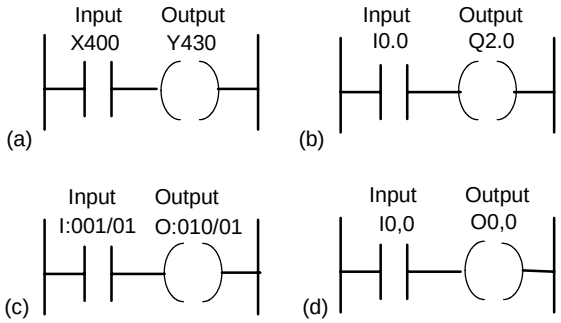


Figure 5.6 *Notation: (a) Mitsubishi, (b) Siemens, (c) Allen-Bradley, (d) Telemecanique*

5.2 Logic functions

There are many control situations requiring actions to be initiated when a certain combination of conditions is realised. Thus, for an automatic drilling machine (as illustrated in Figure 1.1(a)), there might be the condition that the drill motor is to be activated when the limit switches are activated that indicate the presence of the workpiece and the drill position as being at the surface of the workpiece. Such a situation involves the AND logic function, condition A and condition B having both to be realised for an output to occur. This section is a consideration of such logic functions.

5.2.1 AND

Figure 5.7(a) shows a situation where an output is not energised unless two, normally open, switches are both closed. Switch A and switch B have both to be closed, which thus gives an AND logic situation. We can think of this as representing a control system with two inputs A and B (Figure 5.7(b)). Only when A and B are both on is there an output. Thus if we use 1 to indicate an on signal and 0 to represent an off signal, then for there to be a 1 output we must have A and B both 1. Such an operation is said to be controlled by a *logic gate* and the relationship between the inputs to a

logic gate and the outputs is tabulated in a form known as a *truth table*. Thus for the AND gate we have:

Inputs		Output
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

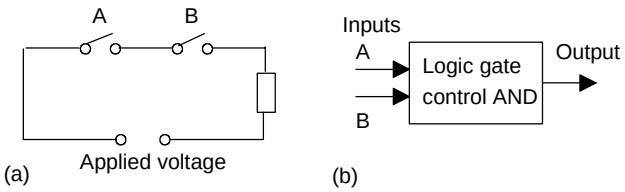


Figure 5.7 *(a) AND circuit, (b) AND logic gate*

An example of an AND gate is an interlock control system for a machine tool so that it can only be operated when the safety guard is in position and the power switched on.

Figure 5.8(a) shows an AND gate system on a ladder diagram. The ladder diagram starts with | |, a normally open set of contacts labelled input A, to represent switch A and in series with it | |, another normally open set of contacts labelled input B, to represent switch B. The line then terminates with O to represent the output. For there to be an output, both input A and input B have to occur, i.e. input A and input B contacts have to be closed (Figure 5.8(b)). In general:

> *On a ladder diagram contacts in a horizontal rung, i.e. contacts in series, represent the logical AND operations*.

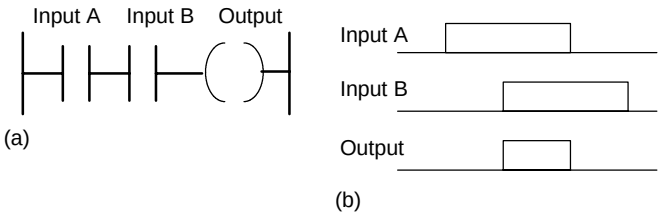


Figure 5.8 *AND gate with a ladder diagram rung*

5.2.2 OR

Figure 5.9(a) shows an electrical circuit where an output is energised when switch A or B, both normally open, are closed. This describes an OR logic gate (Figure 5.9(b)) in that input A or input B must be on for there to be an output. The truth table is:

Inputs		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

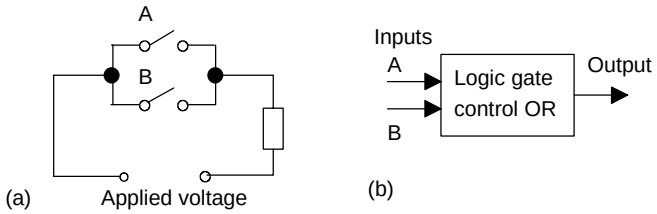


Figure 5.9 *(a) OR electrical circuit, (b) OR logic gate*

Figure 5.10(a) shows an OR logic gate system on a ladder diagram, Figure 5.10(b) showing an equivalent alternative way of drawing the same diagram. The ladder diagram starts with | |, normally open contacts labelled input A, to represent switch A and in parallel with it | |, normally open contacts labelled input B, to represent switch B. Either input A or input B have to be closed for the output to be energised (Figure 5.10(c)). The line then terminates with O to represent the output. In general:

> *Alternative paths provided by vertical paths from the main rung of a ladder diagram, i.e. paths in parallel, represent logical OR operations.*

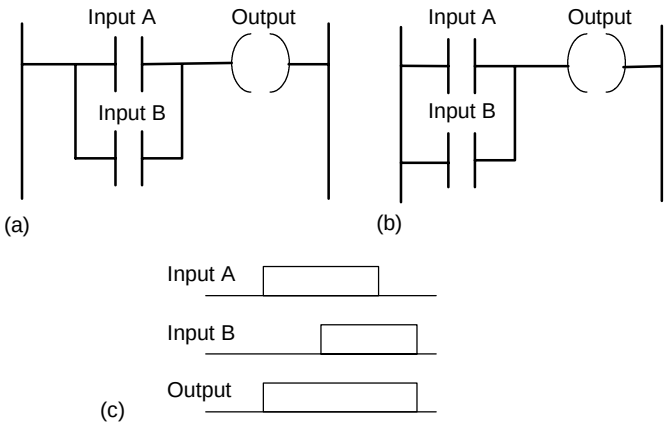


Figure 5.10 *OR gate*

An example of an OR gate control system is a conveyor belt transporting bottled products to packaging where a deflector plate is activated to deflect bottles into a reject bin if either the weight is not within certain tolerances or there is no cap on the bottle.

5.2.3 NOT

Figure 5.11(a) shows an electrical circuit controlled by a switch that is normally closed. When there is an input to the switch, it opens and there is then no current in the circuit. This illustrates a NOT gate in that there is an output when there is no input and no output when there is an input (Figure 5.11(c)). The gate is sometimes referred to as an *inverter*. The truth table is:

Input A	Output
0	1
1	0

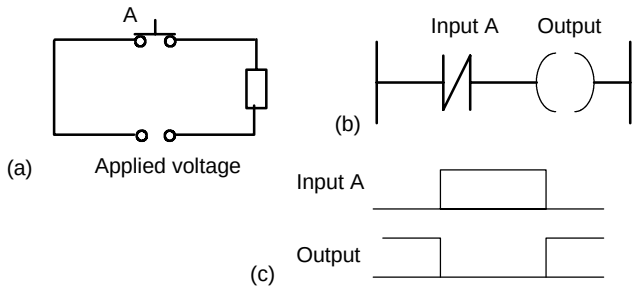


Figure 5.11 *(a) NOT circuit, (b) NOT logic with a ladder rung, (c) high output when no input to A*

Figure 5.11(b) shows a NOT gate system on a ladder diagram. The input A contacts are shown as being normally closed. This is in series with the output ( ). With no input to input A, the contacts are closed and so there is an output. When there is an input to input A, it opens and there is then no output.

An example of a NOT gate control system is a light that comes on when it becomes dark, i.e. when there is no light input to the light sensor there is an output.

5.2.4 NAND

Suppose we follow an AND gate with a NOT gate (Figure 5.12(a)). The consequence of having the NOT gate is to invert all the outputs from the AND gate. An alternative, which gives exactly the same results, is to put a NOT gate on each input and then follow that with OR (Figure 5.12(b)). The same truth table occurs, namely:

Inputs		Output
A	B	
0	0	1
0	1	1
1	0	1
1	1	0

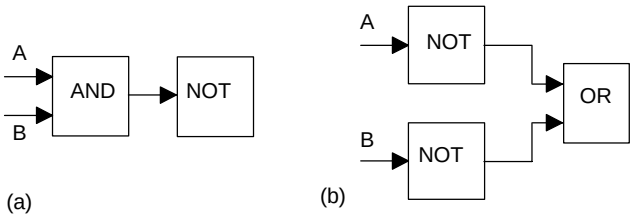


Figure 5.12 *NAND gate*

Both the inputs A and B have to be 0 for there to be a 1 output. There is an output when input A and input B are not 1. The combination of these gates is termed a NAND gate.

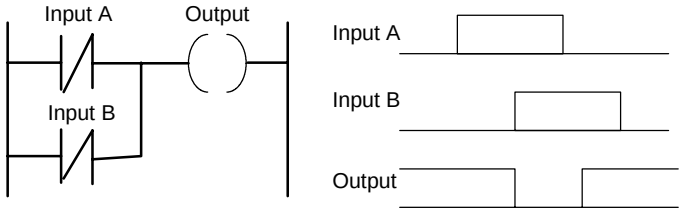


Figure 5.13 *A NAND gate*

Figure 4.18 shows a ladder diagram which gives a NAND gate. When the inputs to input A and input B are both 0 then the output is 1. When the inputs to input A and input B are both 1, or one is 0 and the other 1, then the output is 0.

An example of a NAND gate control system is a warning light that comes on if, with a machine tool, the safety guard switch has not been activated and the limit switch signalling the presence of the workpiece has not been activated.

5.2.5 NOR

Suppose we follow an OR gate by a NOT gate (Figure 5.14(a)). The consequence of having the NOT gate is to invert the outputs of the OR gate. An alternative, which gives exactly the same results, is to put a NOT gate on each input and then an AND gate for the resulting inverted inputs (Figure 5.14(b)). The following is the resulting truth table:

Inputs		Output
A	B	
0	0	1
0	1	0
1	0	0
1	1	0

The combination of OR and NOT gates is termed a NOR gate. There is an output when neither input A or input B is 1.

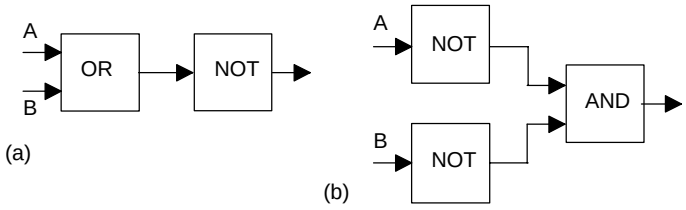


Figure 5.14 *NOR gate*

Figure 5.15 shows a ladder diagram of a NOR system. When input A and input B are both not activated, there is a 1 output. When either X400 or X401 are 1 there is a 0 output.

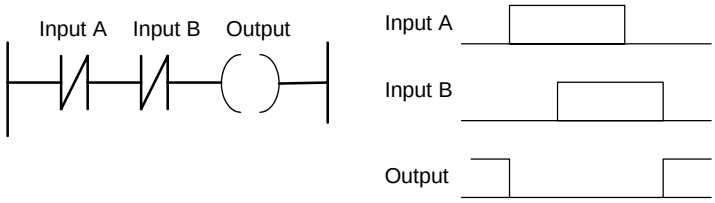


Figure 5.15 *NOR gate*

5.2.6 Exclusive OR (XOR)

The OR gate gives an output when either or both of the inputs are 1. Sometimes there is, however, a need for a gate that gives an output when either of the inputs is 1 but not when both are 1, i.e. has the truth table:

Inputs		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Such a gate is called an *Exclusive* OR or XOR gate. One way of obtaining such a gate is by using NOT, AND and OR gates as shown in Figure 5.16.

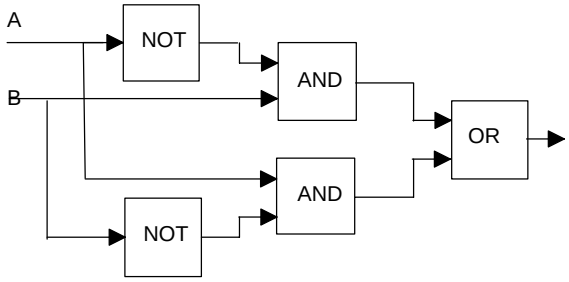


Figure 5.16 *XOR gate*

Figure 5.17 shows a ladder diagram for an XOR gate system. When input A and input B are not activated then there is 0 output. When just input A is activated, then the upper branch results in the output being 1. When just input B is activated, then the lower branch results in the output being 1. When both input A and input B are activated, there is no output. In this example of a logic gate, input A and input B have two sets of contacts in the circuits, one set being normally open and the other normally closed. With PLC programming, each input may have as many sets of contacts as necessary.

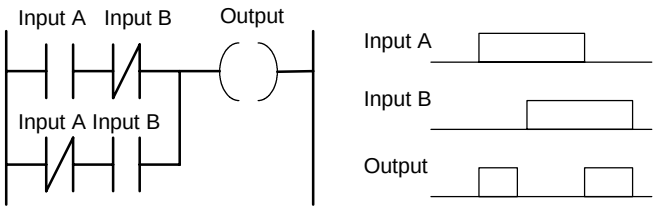


Figure 5.17 *XOR gate*

5.3 Latching

There are often situations where it is necessary to hold an output energised, even when the input ceases. A simple example of such a situation is a motor which is started by pressing a push button switch. Though the switch contacts do not remain closed, the motor is required to continue running until a stop push button switch is pressed. The term *latch circuit* is used for the circuit used to carry out such an operation. It is a self-maintaining circuit in that, after being energised, it maintains that state until another input is received.

An example of a latch circuit is shown in Figure 5.18. When the input A contacts close, there is an output. However, when there is an output, another set of contacts associated with the output closes. These contacts form an OR logic gate system with the input contacts. Thus, even if the input A opens, the circuit will still maintain the output energised. The only way to release the output is by operating the normally closed contact B.

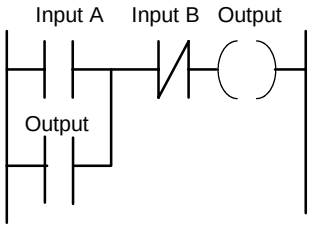


Figure 5.18 *Latched circuit*

As an illustration of the application of a latching circuit, consider a motor controlled by stop and start push button switches and for which one signal light must be illuminated when the power is applied to the motor and another when it is not applied. Figure 5.19 shows the ladder diagram with Mitsubishi notation for the addresses.

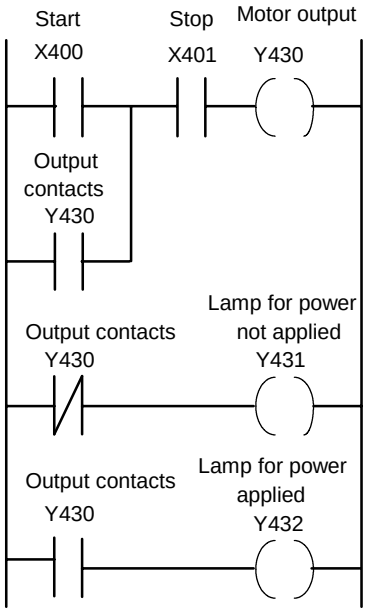


Figure 5.19 *Motor on-off, with signal lamps, ladder diagram. Note that the stop contacts X401 are shown as being programmed as open. If the stop switch used is normally closed then X401 receives a start-up signal to close. This gives a safer operation than programming X401 as normally closed.*

X401 is closed when the program is started. When X400 is momentarily closed, Y430 is energised and its contacts close. This results in latching and also the switching off of Y431 and the switching on of Y432. To switch the motor off, X401 is pressed and opens. Y430 contacts open in the top rung and third rung, but close in the second rung. Thus Y431 comes on and Y432 off.

Latching is widely used with start-ups so that the initial switch on of an application becomes latched on.

5.4 Multiple outputs

With ladder diagrams, there can be more than one output connected to a contact. Figure 5.20 shows a ladder program with two output coils. When the input contacts close both the coils give outputs.

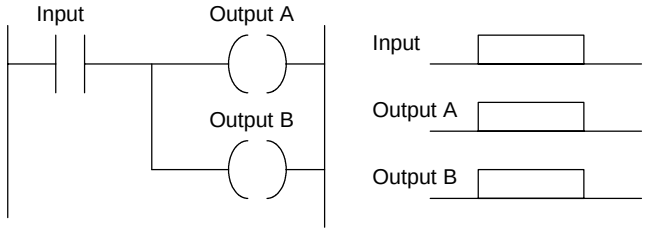


Figure 5.20 *Ladder rung with two outputs*

For the ladder rung shown in Figure 5.21, output A occurs when input A occurs. Output B only occurs when both input A and input B occur.

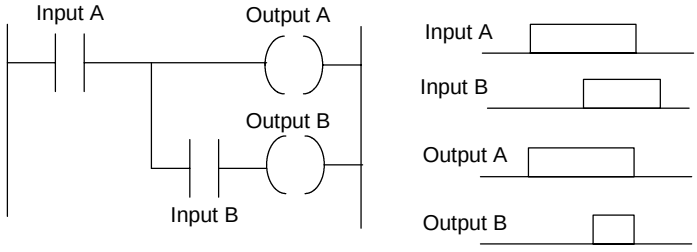
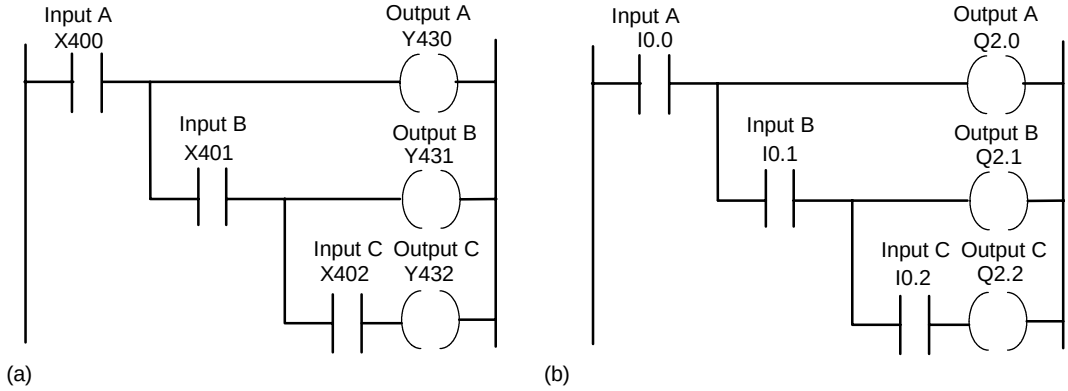


Figure 5.21 *Ladder rung with two inputs and two outputs*

Such an arrangement enables a sequence of outputs to be produced, the sequence being in the sequence with which contacts are closed. Figure 5.22 illustrates this with the same ladder program in Mitsubishi and Siemens notations. Outputs A, B and C are switched on as the contacts in the sequence given by the contacts A, B and C are being closed. Until input A is closed, none of the other outputs can be switched on. When input A is closed, output A is switched on. Then, when input B is closed, output B is switched on. Finally, when input C is closed, output C is switched on.



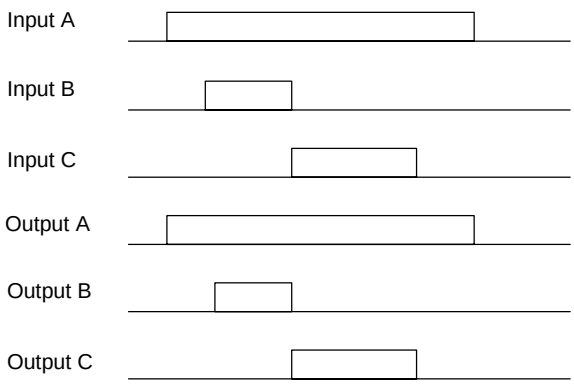


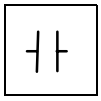
Figure 5.22 *Sequenced outputs*

5.5 Entering programs

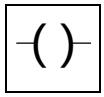
Each horizontal rung on the ladder represents an instruction in the program to be used by the PLC. The entire ladder gives the complete program. There are several methods that can be used for keying in the program into a programming terminal. Whatever method is used to enter the program into a programming terminal or computer, the output to the memory of the PLC has to be in a form that can be handled by the microprocessor. This is termed *machine language* and is just binary code, e.g. 0010100001110001.

5.5.1 Ladder symbols

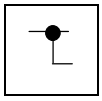
One method of entering the program into the programming terminal involves using a keypad having keys with symbols depicting the various elements of the ladder diagram and keying them in so that the ladder diagram appears on the screen of the programming terminal. For example, to enter a pair of contacts the key marked



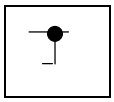
might be used, followed by its address being keyed in. To enter an output the key marked



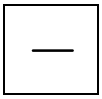
might be used, followed by its address. To indicate the start of a junction



might be pressed; to indicate the end of a junction path



To indicate horizontal circuit links, the following key might be used:



The terminal then translates the program drawn on the screen into machine language.

Computers can be used to draw up a ladder program. These involve loading the computer with the relevant software, e.g. RSLogix from Rockwell Automation Inc. for Allen-Bradley PLCs, MELSOFT – GX Developer for Mitsubishi PLCs, STEP 7 - Micro/WIN V4 for Siemens PLCs. The software operates on the Windows operating system and involves selecting items, in the usual Windows manner, from pull-down menus on the screen.

5.6 Function blocks

The term *function block diagram (FBD)* is used for PLC programs described in terms of graphical blocks. It is described as being a graphical language for depicting signal and data flows through blocks, these being reusable software elements. A function block is a program instruction unit which, when executed, yields one or more output values. Thus a block is represented in the manner shown in Figure 5.23 with the function name written in the box.

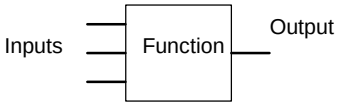


Figure 5.23 *Function block*

The IEC 113-3 standard for drawing such blocks is shown in Figure 5.24. A function block is depicted as a rectangular block with inputs entering from the left and outputs emerging from the right. The function block type name is shown in the block, e.g. AND, with the name of the function block in the system shown above it, Timer1. Names of function block inputs are shown within the block at the appropriate input and output points. Cross diagram connectors are used to indicate where graphical lines would be difficult to draw without cluttering up or complicating a diagram and show where an output at one point is used as an input at another.

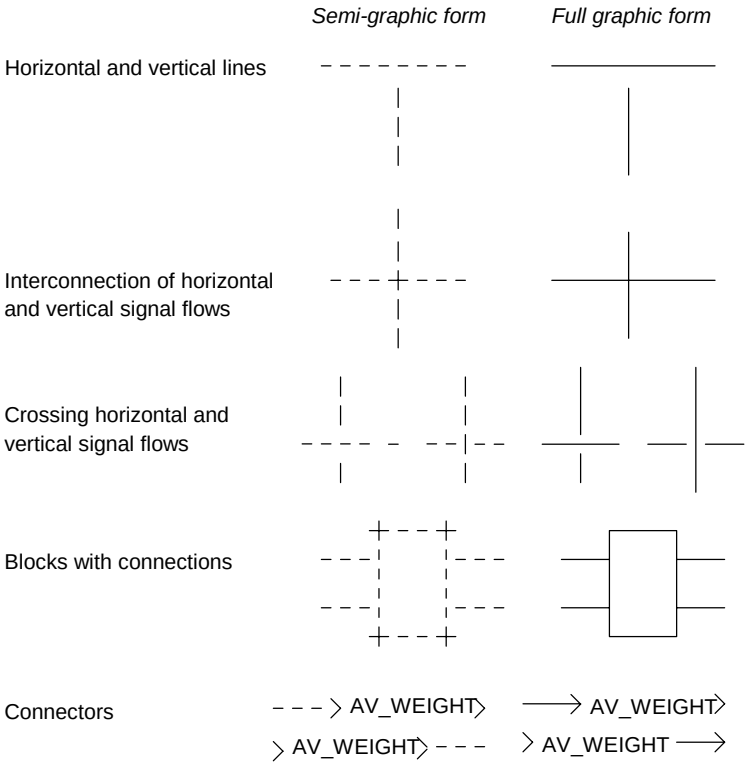


Figure 5.24 *Function block diagrams representation*

Function blocks can have standard functions, such as those of the logic gates or counters or times, or have functions defined by the user, e.g. a block to obtain an average value of inputs.

5.6.1 Logic gates

Programs are often concerned with logic gates. Two forms of standard circuit symbols are used for logic gates, one having originated in the United States and the other being an international standard form (IEEE/ANSI) which uses a rectangle with the logic function written inside it. The 1 in a box indicates that there is an output when the input is 1. The OR function is given by ≥1, this is because there is an output if an input is greater than or equal to 1. A negated input is represented by a small circle on the input, a negative output by a small circle on the output (Figure 5.25). Figure 5.26 shows the symbols. In FBD diagrams the notation used in the IEEE/ANSI form is often encountered.

Figure 5.27 shows the effect of such functional blocks in PLC programs.

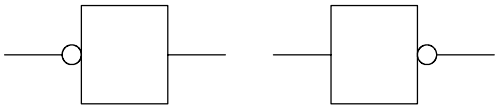


Figure 5.25 *(a) Negated input, (b) negated output*

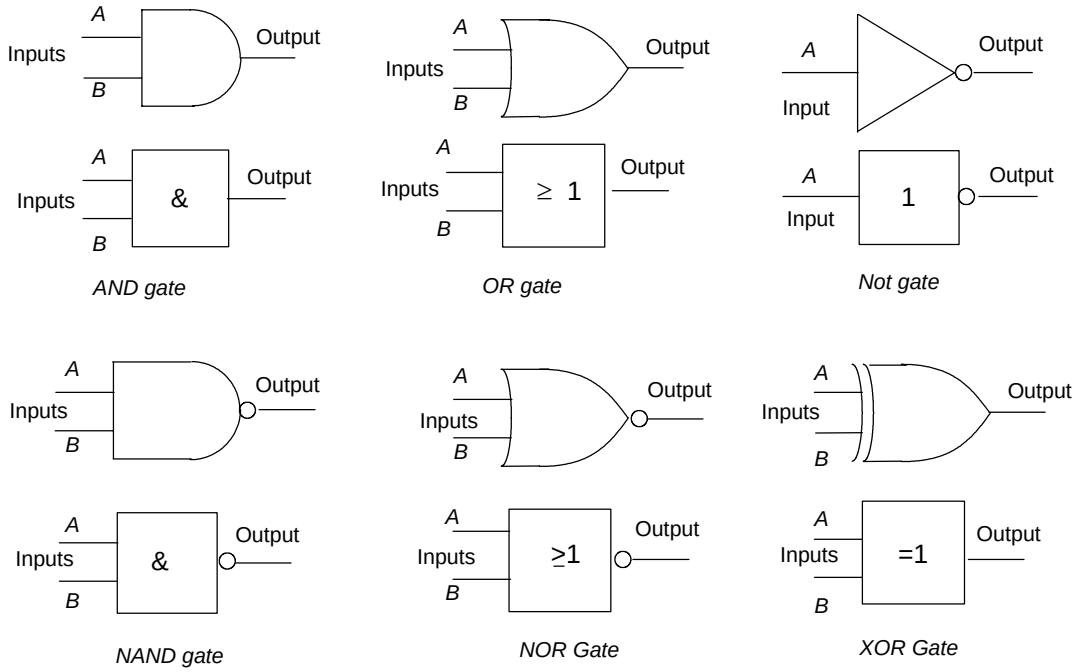


Figure 5.26 *Logic gate symbols*

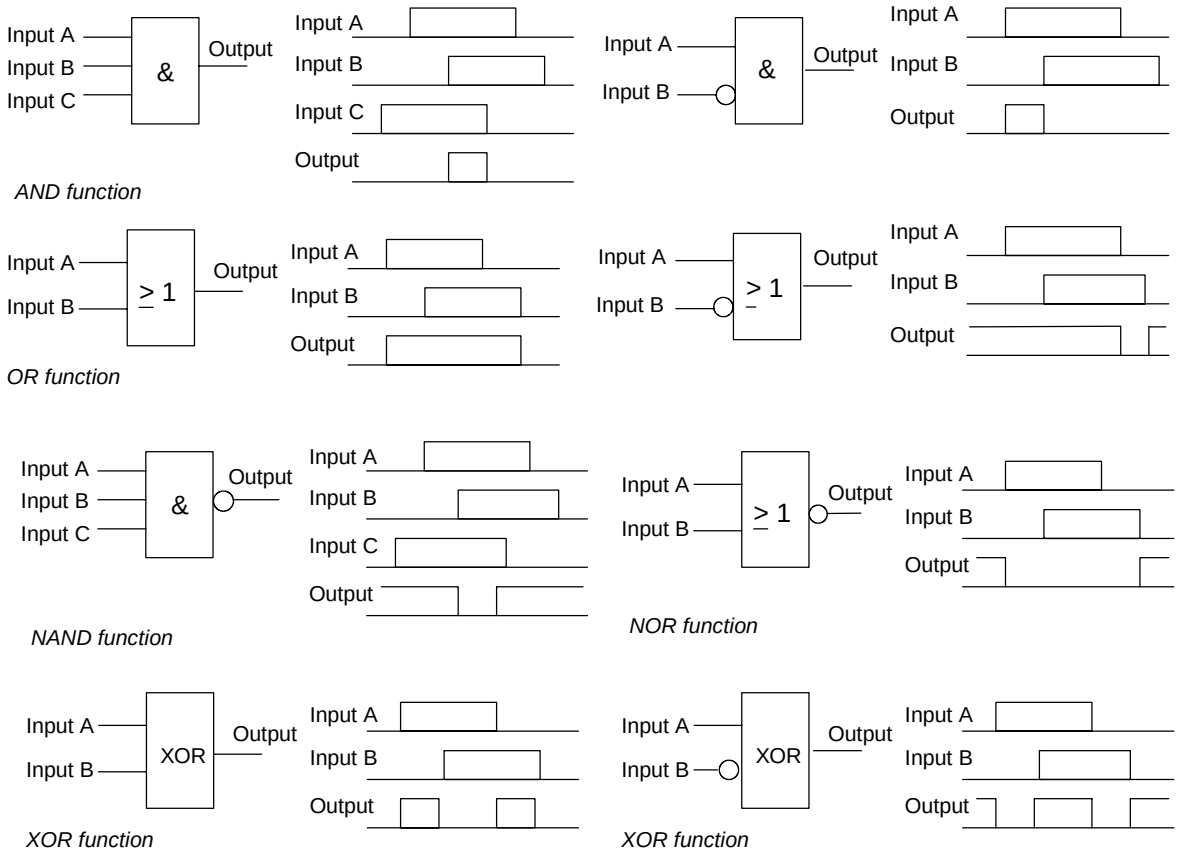


Figure 5.27 *Functional blocks*

To illustrate the form of such a diagram and its relationship to the ladder diagram, Figure 5.28 shows an OR gate. When A or B inputs are 1 then there is an output.

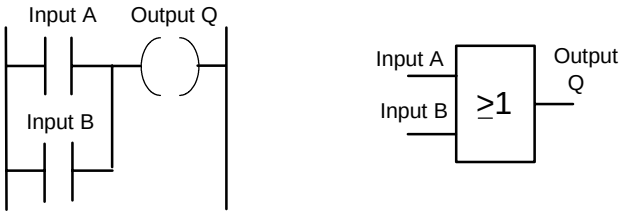


Figure 5.28 *Ladder diagram and equivalent functional block diagram*

Figure 5.29 shows a ladder diagram and its function block equivalent in Siemens notation. The = block is used to indicate an output from the system.

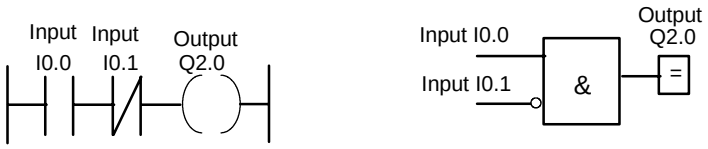


Figure 5.29 *Ladder diagram and equivalent function block diagram*

Figure 5.30 shows a ladder diagram involving the output having contacts acting as an input. The function block diagram equivalent can be shown as a feedback loop.

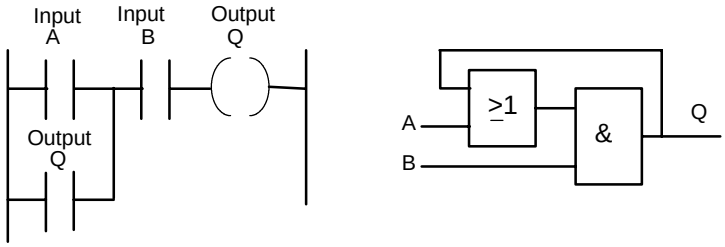


Figure 5.30 *Ladder diagram and equivalent function block diagram*

Consider the development of a function block diagram and ladder diagram for an application in which a pump is required to be activated and pump liquid into a tank when the start switch is closed, the level of liquid in the tank is below the required level and there is liquid in the reservoir from which it is to be pumped. What is required is an AND logic situation between the start switch input and a sensor input which is on when the liquid in the tank is below the required level. We might have a switch which is on until the liquid is at the required level. These two elements are then in an AND logic situation with a switch indicating that there is liquid in the reservoir. Suppose this switch gives an input when there is liquid. The function block diagram, and the equivalent ladder diagram, is then of the form shown in Figure 5.31.

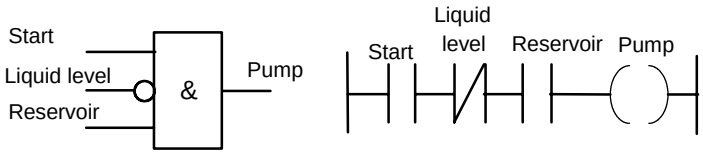


Figure 5.31 *Pump application*

5.6.2 Boolean algebra

Ladder programs can be derived from Boolean expressions since we are concerned with a mathematical system of logic. In Boolean algebra there are just two digits, 0 and 1. When we have an AND operation for inputs A and B then we can write:

$$A \cdot B = Q$$

where Q is the output. Thus Q is equal to 1 only when A = 1 and B = 1. The OR operation for inputs A and B is written as:

$$A + B = Q$$

Thus Q is equal to 1 only when A = 1 or B = 1. The NOT operation for an input A is written as:

$$\overline{A} = Q$$

Thus when A is not 1 there is an output.

As an illustration of how we can relate Boolean expressions with ladder diagrams, consider the expression:

$$A + B \cdot C = Q$$

This tells us that we have A or the term B and C giving the output Q. Figure 5.32 shows the ladder and functional block diagrams. Written in terms of Mitsubishi notation, the above expression might be:

$$X400 + X401 \cdot X402 = Y430$$

In Siemens notation it might be:

$$I0.0 + I0.1 \cdot I0.2 = Q2.0$$

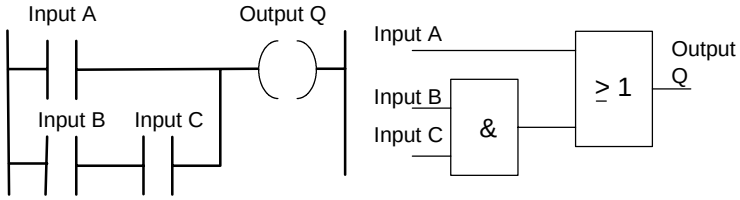


Figure 5.32 *Ladder diagram*

As a further illustration, consider the Boolean expression:

$$A + \overline{B} = Q$$

Figure 5.33 shows the ladder and functional block diagrams.

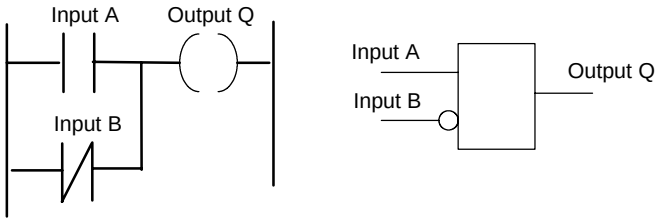


Figure 5.33 *Ladder diagram*

Written in terms of Mitsubishi notation, the expression might be:

$$X400 + \overline{X401} = Y430$$

and in Siemens notation:

$$I0.0 + \overline{I0.1} = Q2.0$$

Consider the exclusive-OR gate and its assembly from NOT, AND and OR gates, as shown in Figure 5.34.

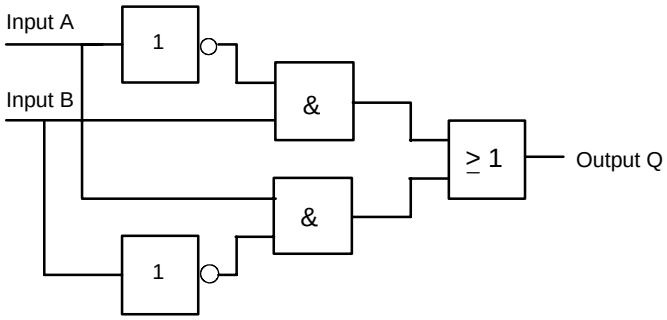


Figure 5.34 *XOR gate*

The input to the bottom AND gate is:

$$A \text{ and } \overline{B}$$

and so its output is:

$$A \cdot \overline{B}$$

The input to the top AND gate is:

$$\overline{A} \text{ and } B$$

and so its output is:

$$\overline{A} \cdot B$$

Thus the Boolean expression for the output from the OR gate is:

$$A \cdot \overline{B} + \overline{A} \cdot B = Q$$

Consider a logic diagram with many inputs, as shown in Figure 5.35, and its representation by a Boolean expression and a ladder rung.

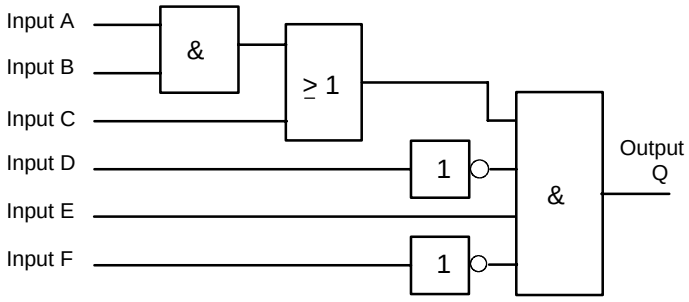


Figure 5.35 *Logic diagram*

For inputs A and B we obtain an output from the upper AND gate of A·B. From the OR gate we obtain an output of A·B + C. From the lower AND gate we obtain an output Q of:

$$(A \cdot B + C) \cdot \overline{D} \cdot E \cdot \overline{F} = Q$$

The ladder diagram to represent this is shown in Figure 5.36.

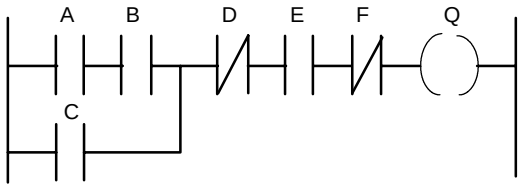


Figure 5.36 *Ladder diagram for Figure 5.35*

5.7 Program examples

The following tasks are intended to illustrate the application of the programming techniques given in this chapter.

A signal lamp is required to be switched on if a pump is running and the pressure is satisfactory, or if the lamp test switch is closed. For the inputs from the pump and the pressure sensors we have an AND logic situation since both are required if there is to be an output from the lamp. We, however, have an OR logic situation with the test switch in that it is required to give an output of lamp on regardless of whether there is a signal from the AND system. The function block diagram and the ladder diagram are thus of the form shown in Figure 5.37. Note that with the ladder diagram we tell the PLC when it has reached the end of the program by the use of the END or RET instruction.

As another example, consider a valve which is to be operated to lift a load when a pump is running and either the lift switch is operated or a switch operated indicating that the load has not already been lifted and is at the bottom of its lift channel. We have an OR situation for the two switches and an AND situation involving the two switches and the pump. Figure 5.38 shows a possible program.

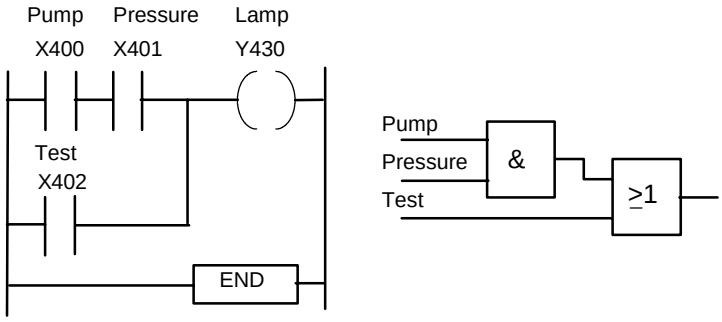


Figure 5.37 *Signal lamp task*

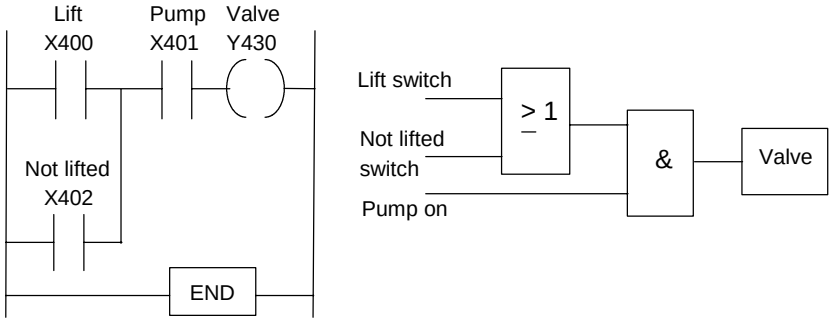


Figure 5.38 *Valve operation program*

As another example, consider a system where there has to be no output when any one of four sensors gives an output, otherwise there is to be an output. One way we could write a program for this is for each sensor to have contacts that are normally closed so there is an output. When there is an input to the sensor the contacts open and the output stops. We have an AND logic situation. Figure 5.39 shows the functional block diagram and the ladder diagram of a system that might be used.

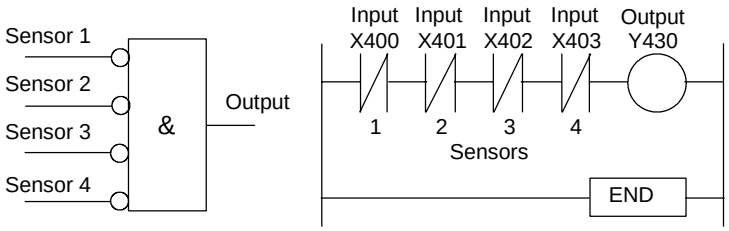
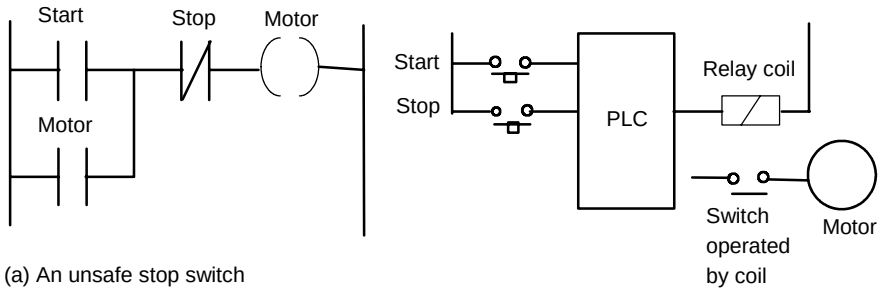


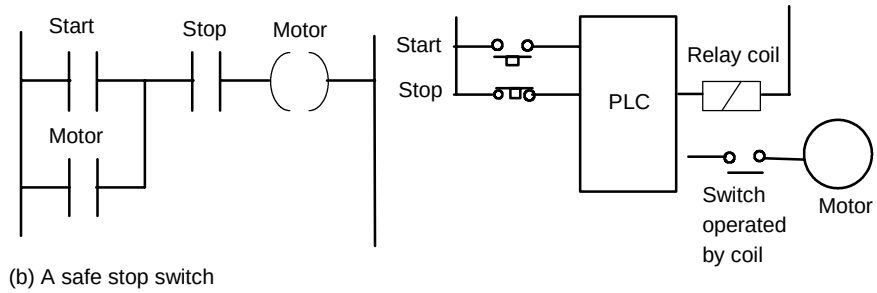
Figure 5.39 *Output switched off by any one of four sensors being activated*

5.7.1 Location of stop switches

The location of stop switches with many applications has to be very carefully considered in order to ensure a safe system. A stop switch is *not* safe if it is normally closed and has to be opened to give the stop action. If the switch malfunctions and remains closed then the system cannot be stopped. Figure 5.40(a) illustrates this. A better arrangement is to program the stop switch in the ladder program as open in Figure 5.33(b) and use a stop switch that is normally closed and operating opens it. Thus there is an input signal to the system which closes the contacts in the program when it starts up.



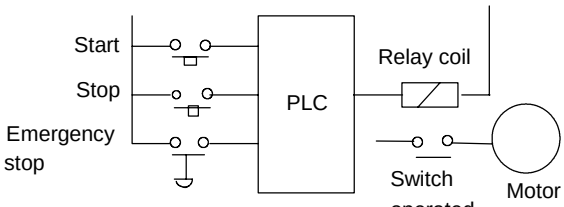
(a) An unsafe stop switch



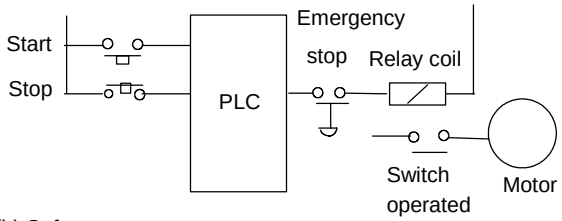
(b) A safe stop switch

Figure 5.40 *Motor stop switch location*

Figure 5.41 shows where we can safely locate an emergency stop switch. If it is in the input to the PLC (Figure 5.41(a)) then if the PLC malfunctions it may not be possible to stop the motor. However, if the emergency stop switch is in the output, operating it will stop the motor and also cause the start switch to become unlatched if the arrangement shown in Figure 5.41(b) is being used. The motor will thus not restart when the emergency stop button is released.



(a) Unsafe arrangement



(b) Safe arrangement

Figure 5.41 *Location of emergency stop switch*

Problems

Questions 1 to 19 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

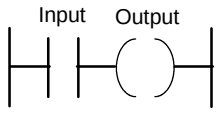


Figure 5.42 *Problem 1*

1 Decide whether each of these statements is True (T) or False (F).

Figure 5.42 shows a ladder diagram rung for which:
(i) The input contacts are normally open.
(ii) There is an output when there is an input to the contacts.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

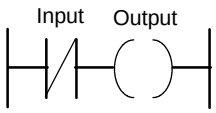


Figure 5.43 *Problem 2*

2 Decide whether each of these statements is True (T) or False (F).

Figure 5.43 shows a ladder diagram rung for which:
(i) The input contacts are normally open.
(ii) There is an output when there is an input to the contacts.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

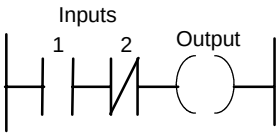


Figure 5.44 *Problem 3*

3 Decide whether each of these statements is True (T) or False (F).

Figure 5.44 shows a ladder diagram rung for which:
(i) When only input 1 contacts are activated, there is an output.
(ii) When only input 2 contacts are activated, there is an output.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

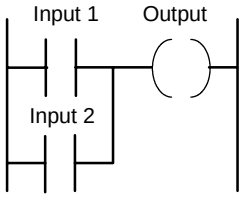


Figure 5.45 *Problem 4*

4 Decide whether each of these statements is True (T) or False (F).

Figure 5.45 shows a ladder diagram rung for which there is an output when:
(i) Inputs 1 and 2 are both activated.
(ii) Either one of inputs 1 and 2 is not activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

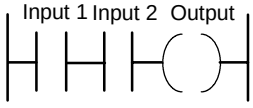


Figure 5.46 *Problem 5*

5 Decide whether each of these statements is True (T) or False (F).

Figure 5.46 shows a ladder diagram rung with an output when:
(i) Inputs 1 and 2 are both activated.
(ii) Input 1 or 2 is activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

6 Decide whether each of these statements is True (T) or False (F).

Figure 5.47 shows a ladder diagram rung for which there is an output when:
(i) Input 1 is momentarily activated before reverting to its normally open state.
(ii) Input 2 is activated.

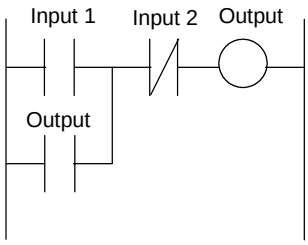


Figure 5.47 *Problem 6*

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Questions 7 to 10 refer to the following logic gate systems:

A AND
B OR
C NOR
D NAND

7 Which form of logic gate system is given by a ladder diagram with a rung having two normally open sets of contacts in parallel?

8 Which form of logic gate system is given by a ladder diagram with a rung having two normally closed gates in parallel?

9 Which form of logic gate system is given by a ladder diagram with a rung having two normally closed gates in series?

10 Which form of logic gate system is given by a ladder diagram with a rung having two normally open gates in series?

Problems 11 to 14 concern Boolean expressions for inputs A and B.

A Input A is in series with input B, both inputs being normally off.
B Input A is in parallel with input B, both inputs being normally off.
C Input A, normally off, is in series with input B which is normally on.
D Input A is in parallel with input B, both inputs being normally on.

11 Which arrangement of inputs is described by the Boolean relationship $A \cdot B$?

12 Which arrangement of inputs is described by the Boolean relationship $A + B$?

13 Which arrangement of inputs is described by the Boolean relationship $\overline{A} + \overline{B}$?

14 Which arrangement of inputs is described by the Boolean relationship $A \cdot \overline{B}$?

15 The arrangement of inputs in Figure 5.48 is described by the Boolean expression:

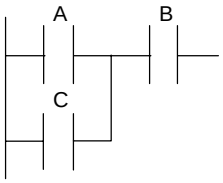


Figure 5.48 *Problem 15*

A A·B·C
B (A + C)·B
C (A + B)·C
D A·C + B

16 Decide whether each of these statements is True (T) or False (F).

For the function block diagram in Figure 5.49, there is an output:
(i) When A is 1.
(ii) When B is 1.

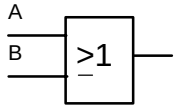


Figure 5.49 *Problem 16*

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

17 Decide whether each of these statements is True (T) or False (F).

For the function block diagram in Figure 5.50, there is an output:
(i) When A is 1.
(ii) When B is 1.

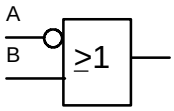


Figure 5.50 *Problem 17*

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

18 Decide whether each of these statements is True (T) or False (F).

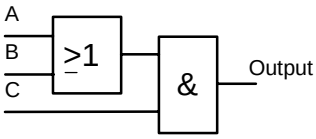


Figure 5.51 *Problem 18*

For the functional block diagram in Figure 5.51, there is an output:
(i) When A is 1, B is 0 and C is 0.
(ii) When A is 0, B is 1 and C is 1.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

19 Decide whether each of these statements is True (T) or False (F).

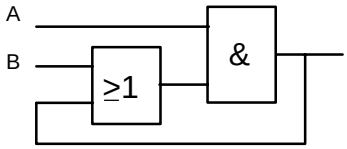


Figure 5.52 *Problem 19*

For the function block diagram in Figure 5.52, with A being a steady input condition and B a momentary input, there is an output:
(i) When A is 1 and B is 0.
(ii) When A is 0 and B is 1.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

20 Figure 5.53(a) shows a ladder diagram. Which of the function block diagrams in Figure 5.53(b) is its equivalent?

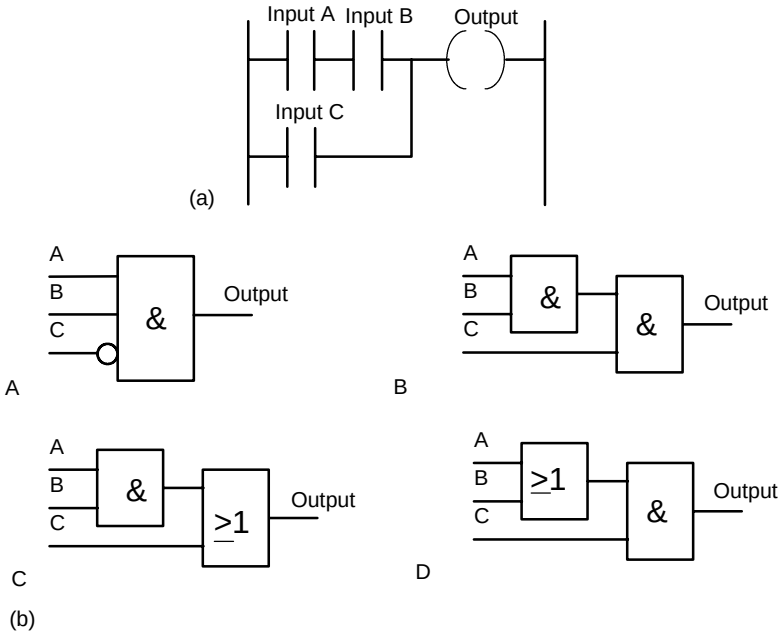


Figure 5.53 *Problem 20*

21 Figure 5.54(a) shows a function block diagram. Which of the ladder diagrams in Figure 5.54(b) is the equivalent?

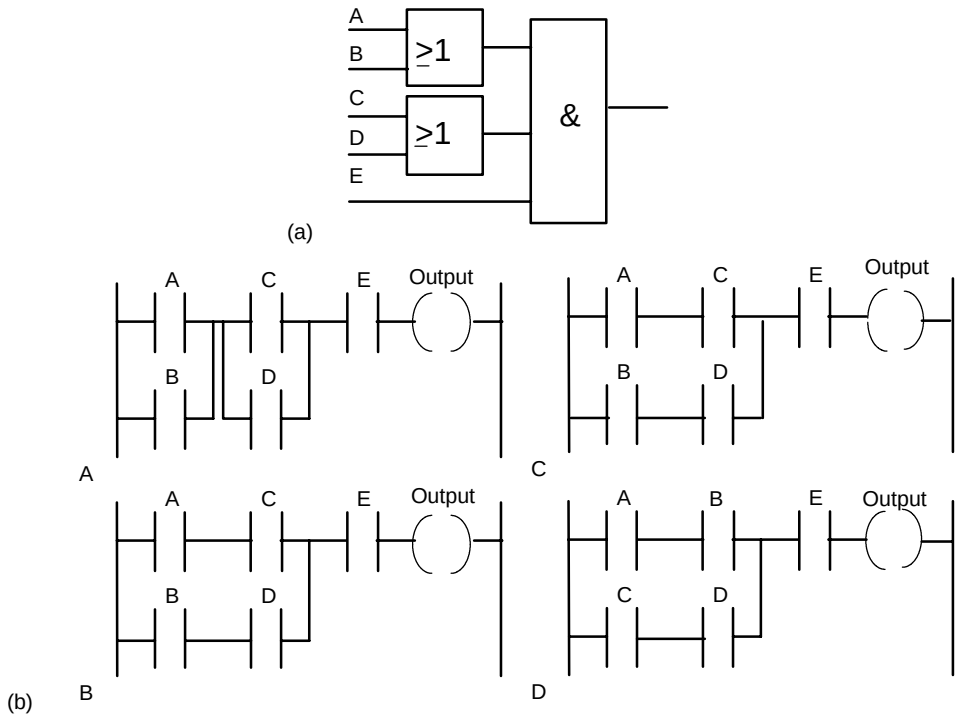


Figure 5.54 *Problem 21*

22 Draw the ladder rungs to represent:
(a) Two switches are normally open and both have to be closed for a motor to operate.
(b) Either of two, normally open, switches have to be closed for a coil to be energised and operate an actuator.
(c) A motor is switched on by pressing a spring-return push button start switch, and the motor remains on until another spring-return push button stop switch is pressed.
(d) A lamp is to be switched on if there is an input from sensor A or sensor B.
(e) A light is to come on if there is no input to a sensor.
(f) A solenoid valve is to be activated if sensor A gives an input.

6 IL, SFC and ST programming methods

This chapter continues from the previous chapter and discusses the other IEC 1131-3 (International Electrotechnical Commission) programming languages, i.e. instruction lists (IL), sequential function charts (SFC), and structured text (ST).

6.1 Instruction lists

A programming method, which can be considered to be the entering of a ladder program using text, is *instruction lists (IL)*. Instruction list gives programs which consist of a series of instructions, each instruction being on a new line. An instruction consists of an operator followed by one of more operands, i.e. the subjects of the operator. In terms of ladder diagrams an operator may be regarded as a ladder element. Each instruction may either use or change the value stored in a memory register. For this, mnemonic codes are used, each code corresponding to an operator/ladder element. The codes used differ to some extent from manufacturer to manufacturer, though a standard IEC 1131-3 has been proposed and is being widely adopted. Table 6.1 shows some of the codes used by manufacturers, and the proposed standard, for instructions used in this chapter (see later chapters for codes for other functions).

Table 6.1 *Instruction code mnemonics*

IEC 1131-3 Operators	Mitsubishi	OMRON	Siemens/ Telemecanique	Operation	Ladder diagram
LD	LD	LD	A	Load operand into result register	Start a rung with open contacts
LDN	LDI	LD NOT	AN	Load negative operand into result register	Start a rung with closed contacts
AND	AND	AND	A	Boolean AND	A series element with open contacts
ANDN	ANI	AND NOT	AN	Boolean AND with negative operand	A series element with closed contacts
OR	OR	OR	O	Boolean OR	A parallel element with open contacts
ORN	ORI	OR NOT	ON	Boolean OR with negative operand	A parallel element with closed contacts
ST	OUT	OUT	=	Store result register into operand	An output

As an illustration of the use of IEC 113-1 operators, consider the following:

```
LD      A       (*Load A*)
AND     B       (*AND B*)
ST      Q       (* Store result in Q, i.e. output to Q*)
```

In the first line of the program, LD is the operator, A the operand, and the words at the ends of program lines and in brackets and preceded and followed by * are comments added to explain what the operation is and are not part of the program operation instructions to the PLC. LD A is thus the instruction to load the A into the memory register. It can then later be called on for further operations. The next line of the program has the Boolean operation AND performed with A and B. The last line has the result stored in Q, i.e. outputted to Q.

Labels can be used to identify various entry points to a program, useful as we will find later for jumps in programs, and these precede the instruction and are separated from it by a colon. Thus we might have:

```
PUMP_OK:    LD          C           (* Load C*)
```

with there being the instruction earlier in the program to jump to PUMP_OK if a particular condition is realised.

With the IEC 113-1 operators, an N after it is used to negate its value. For example, if we have:

```
LD      A       (*Load A*)
ANDN    B       (*AND NOT B*)
```

Thus the ANDN operator inverts the value of ladder contacts and ANDs the result.

6.1.1 Ladder programs and instruction lists

When looked at in terms of ladder diagrams, whenever a rung is started, it must use a ‘start a rung code’. This might be LD, or perhaps A or L, to indicate the rung is starting with open contacts, or LDI, or perhaps LDN or LD NOT or AN or LN, to indicate it is starting with closed contacts. All rungs must end with an output or store result code. This might be OUT or = or ST. The following shows how individual rungs on a ladder are entered using the Mitsubishi mnemonics for the AND gate, shown in Figure 6.1(a).

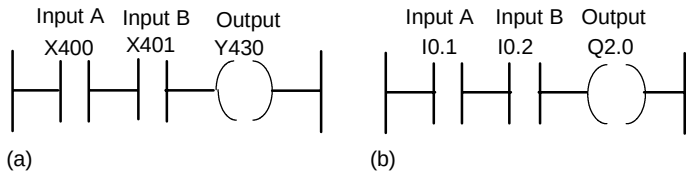


Figure 6.1 *AND gate: (a) Mitsubishi, (b) Siemens*

The rung start with LD because it is starting with open contacts. Since the address of the input is X400, the instruction is LD X400. This is followed by another open contacts input and so the next program line involves the instruction AND with the address of the element, thus the instruction is AND X401. The rung terminates with an output and so the instruction OUT is used with the address of the output, i.e. OUT Y430. The single rung of a ladder would thus be entered as:

```
LD      X400
AND     X401
OUT     Y430
```

For the same rung with Siemens notation (Figure 6.1(b)) we have:

```
A       I0.1
A       I0.2
=       Q2.0
```

Consider another example, an OR gate. Figure 6.2(a) shows the gate with Mitsubishi notation.

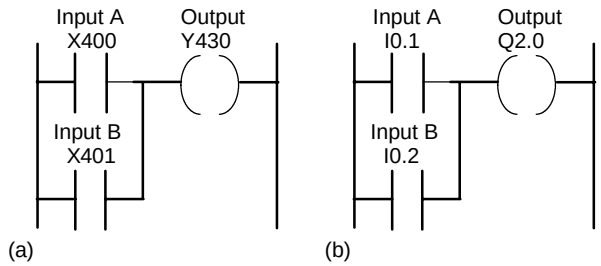


Figure 6.2 *OR gate: (a) Mitsubishi, (b) Siemens*

The instructions for the rung start with an open contact is LD X400. The next item is the parallel OR set of contacts X401. Thus the next instruction is OR X401. The last step is the output, hence OUT Y430. The instruction list would thus be:

```
LD      X400
OR      X401
OUT     Y430
```

Figure 6.2(b) shows the Siemens version of the OR gate. The following is the Siemens instruction list:

```
A       I0.1
O       I0.2
=       Q2.0
```

Figure 6.3(a) shows the ladder system for a NOR gate in Mitsubishi notation.

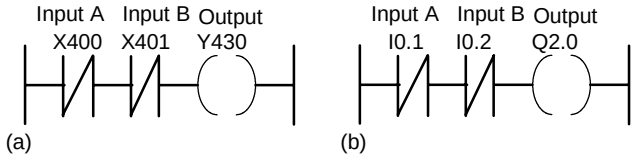


Figure 6.3 *NOR gate: (a) Mitsubishi, (b) Siemens*

The rung starts with normally closed contacts and so the instruction is LDI. I when added to Mitsubishi instruction is used to indicate the inverse of the instruction. The next step is a series normally closed contact and so ANI, again the I being used to make an AND instruction the inverse. I is also the instruction for a NOT gate. The instructions for the NOR gate rung of the ladder would thus be entered as:

```
LDI     X400
ANI     X401
OUT     Y430
```

Figure 6.3(b) shows the NOR gate with Siemens notation. Note that N added to an instruction is used to make the inverse. The instruction list then becomes:

```
LN      I0.1
AN      I0.2
=       Q2.0
```

Consider the rung shown in Figure 6.4(a) in Mitsubishi notation, a NAND gate.

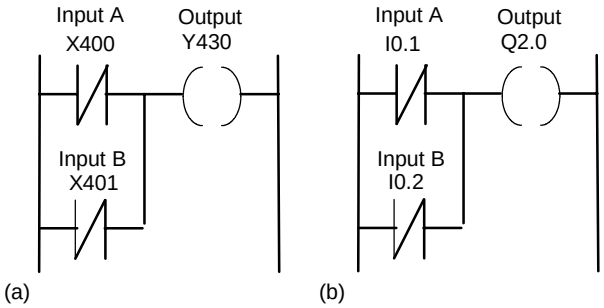


Figure 6.4 *NAND gate: (a) Mitsubishi, (b) Siemens*

It starts with the normally closed contacts X400 and so starts with the instruction LDI X400. The next instruction is for a parallel set of normally closed contacts, thus the instruction is ORI X401. The last step is the output, hence OUT Y430. The instruction list is thus:

```
LDI     X400
ORI     X401
OUT     Y430
```

Figure 6.4(b) shows the NAND gate in Siemens notation. The instruction list is then:

```
AN      I0.1
ON      I0.2
=       Q2.0
```

6.1.2 Branch codes

The Exclusive OR (XOR) gate shown in Figure 6.5(a) in Mitsubishi notation has two parallel arms with an AND situation in each arm.

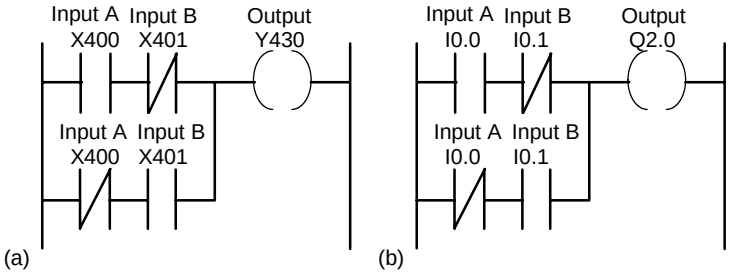


Figure 6.5 *XOR gate: (a) Mitsubishi, (b) Siemens*

In such a situation Mitsubishi uses an ORB instruction to indicate 'OR together parallel branches'. The first instruction is for a normally open pair of contacts X400. The next instruction is for a series set of normally closed contacts X401, hence ANI X401. After reading the first two instructions, the third instruction starts a new line. It is recognised as a new line because it starts with LDI, all new lines starting with LD or LDI. But the first line has not been ended by an output. The PLC thus recognises that a parallel line is involved for the second line and reads together the listed elements until the ORB instruction is reached. The mnemonic ORB (OR branches/blocks together) indicates to the PLC that it should OR the results of the first and second instructions with that of the new branch with the third and fourth instructions. The list concludes with the output OUT Y430. The instruction list would thus be entered as:

```
LD      X400
ANI     X401
LDI     X400
AND     X401
ORB
OUT     Y430
```

Figure 6.5(b) shows the Siemens version of XOR gate. Brackets are used to indicate that certain instructions are to be carried out as a block. They are used in the same way as brackets in any mathematical equation. For example, (2 + 3)/4 means that the 2 and 3 must be added before dividing by 4. Thus with the Siemens instruction list we have in step 0 the instruction A(. The brackets close in step 3. This means that the A in step 0 is applied only after the instructions in steps 1 and 2 have been applied.

Step	Instruction	
0	A(	
1	A	I0.0
2	AN	I0.1
3	)	
4	O(	
5	AN	I0.0
6	A	I0.1
7	)	
8	=	Q2.0

.

The IEC 1131-3 standard for such programming is to use brackets in the way used in the above Siemens example.

Figure 6.6(a) shows a circuit, in Mitsubishi notation, which can be considered as two branched AND blocks. The instruction used here is ANB. The instruction list is thus:

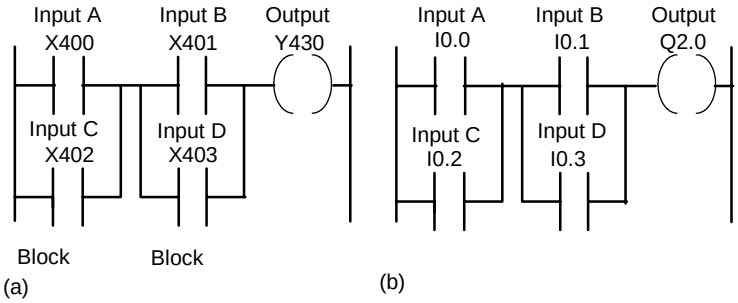


Figure 6.6 *Two branched AND gates: (a) Mitsubishi, (b) Siemens*

Step	Instruction	
0	LD	X400
1	OR	X402
2	LD	X401
3	OR	X403
4	ANB	
5	OUT	Y430

Figure 6.6(b) shows the same circuit in Siemens notation. Such a program is written as an instruction list using brackets. The A instruction in step 0 applies to the result of steps 1 and 2. The A instruction in step 4 applies to the result of steps 5 and 6. The program instruction list is thus:

Step	Instruction	
0	A(	
1	A	I0.0
2	O	I0.2
3	)	
4	A(	

5	A	I0.1
6	O	I0.3
7	)	
8	=	Q2.0

6.1.3 More than one rung

Figure 6.7(a) shows a ladder, in Mitsubishi notation, with two rungs. In writing the instruction list we just write the instructions for each line in turn. The instruction LD or LDI indicates to the PLC that a new rung is starting. The instruction list is thus:

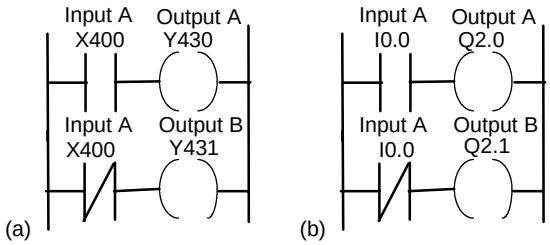


Figure 6.7 *Toggle circuit: (a) Mitsubishi, (b) Siemens*

```
LD      X400
OUT     Y430
LDI     X400
OUT     Y431
```

The system is one where when X400 is not activated, there is an output from Y431 but not Y430. When X400 is activated, there is then an output from Y430 but not Y431.

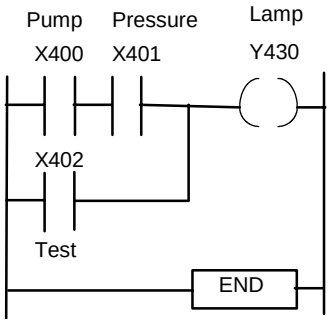
Figure 6.7(b) shows the same program in Siemens notation. The = instruction indicates the end of a line. The A or AN instruction does not necessarily indicate the beginning of a rung since the same instruction is used for AND and AND NOT. The instruction list is then:

```
A       I0.0
=       Q2.0
AN      I0.0
=       Q2.1
```

6.1.4 Programming examples

The following tasks are intended to illustrate the application of the programming techniques given in this section and are the examples for which ladder diagrams and function block diagrams were derived in Section 5.7. See that section for an explanation of the ladder diagrams; here we just show the instruction lists relating to the programs.

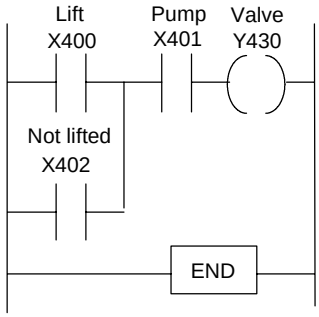
A signal lamp is required to be switched on if a pump is running and the pressure is satisfactory, or if the lamp test switch is closed. Figure 6.8 shows the ladder program and the related instruction list.




```
LD      X400
AND     X401
LD      X402
ORB
OUT     Y430
END
```

Figure 6.8 *Signal lamp task*

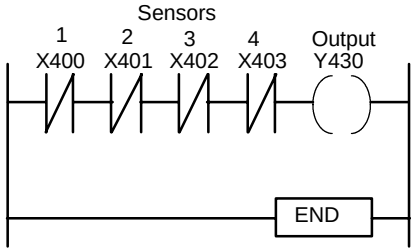
For a valve which is to be operated to lift a load when a pump is running and either the lift switch is operated or a switch operated indicating that the load has not already been lifted and is at the bottom of its lift channel, Figure 6.9 shows the ladder program and the related instruction list.




```
LD      X400
OR      X402
AND     X401
OUT     Y430
END
```

Figure 6.9 *Valve operation program*

For a system where there has to be no output when any one of four sensors gives an output, otherwise there is to be an output, Figure 6.10 shows the ladder program and the instruction list.




```
LDI     X400
ANI     X401
ANI     X402
ANI     X403
OUT     Y430
END
```

Figure 6.10 *Output switched off by any one of four sensors being activated*

6.2 Sequential function charts

If we wanted to describe a traffic lamp sequence, one way we could do this would be to represent it as a sequence of functions or states such as red light state and green light state and the inputs and outputs to each

state. Figure 6.11 illustrates this. State 0 has an input which is triggered after the green light has been on for 1 minute and an output of red light on. State 1 has an input which is triggered after the red light has been on for 1 minute and an output of green light on.

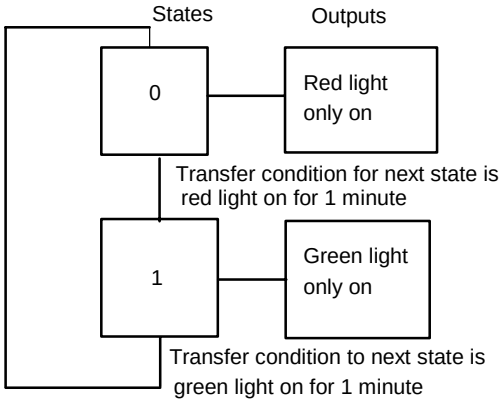

States
Outputs
0
Red light only on
Transfer condition for next state is red light on for 1 minute
1
Green light only on
Transfer condition to next state is green light on for 1 minute

Figure 6.11 *Sequence for traffic lights*

The term *sequential function chart* (SFC) is used for a pictorial representation of a system's operation to show the sequence of the events involved in its operation. SFC charts have the following elements:

1 The operation is described by a number of separate sequentially connected states or steps which are represented by rectangular boxes, each representing a particular state of the system being controlled. The initial step in a program is represented differently to the other steps, Figure 6.12 shows its representation.

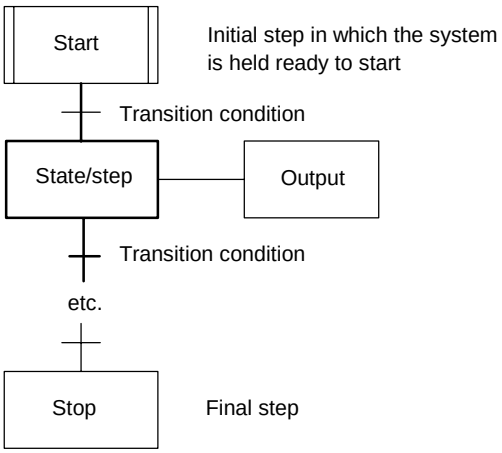


Figure 6.12 *A state and its transition*

2 Each connecting line between states has a horizontal bar representing the transition condition that has to be realised before the system can move from one state to the next. Two steps can never be directly

connected, they must always be separated by a transition. Two transitions can never directly follow from one to another, they must always be separated by a step.

3 When the transfer conditions to the next state are realised then the next state or step in the program occurs.

4 The process thus continues from one state to the next until the complete machine cycle is completed.

5 Outputs/actions at any state are represented by horizontally linked boxes and occur when that state has been realised.

As an illustration, Figure 6.13 shows part of an SFC and its equivalent ladder diagram.

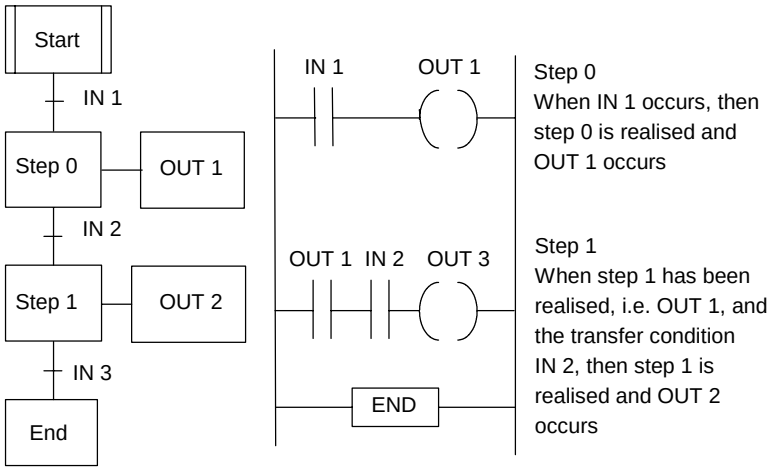

Start
IN 1
Step 0
OUT 1
IN 2
Step 1
OUT 2
IN 3
End
IN 1
OUT 1
Step 0
When IN 1 occurs, then step 0 is realised and OUT 1 occurs
OUT 1 IN 2 OUT 3
Step 1
When step 1 has been realised, i.e. OUT 1, and the transfer condition IN 2, then step 1 is realised and OUT 2 occurs
END

Figure 6.13 *SFC and equivalent ladder diagram*

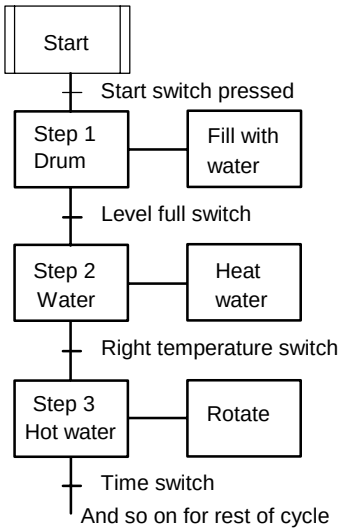


Figure 6.14 *Washing machine*

As an illustration of the principles of SFC, consider the situation with, say, part of the washing cycle of a domestic washing machine where the drum is to be filled with water and then when full a heater has to be switched on and remain on until the temperature reaches the required level. Then the drum is to be rotated for a specified time. We have a sequence of states which can be represented in the manner shown in Figure 6.14.

6.2.1 Branching and convergence

Selective branching is illustrated in Figure 6.15 and allows for different states to be realised depending on the transfer condition that occurs.

Parallel branching (Figure 6.16), represented by a pair of horizontal lines, allows for two or more different states to be realised and proceed simultaneously.

Figures 6.17 and 6.18 show how convergence is represented by an SFC. In Figure 6.17 the sequence can go from state 2 to state 4 if IN 4 occurs or from state 3 to state 4 if IN 5 occurs. In Figure 6.18 the sequence can go simultaneously from both state 2 and state 3 to state 4 if IN 4 occurs.

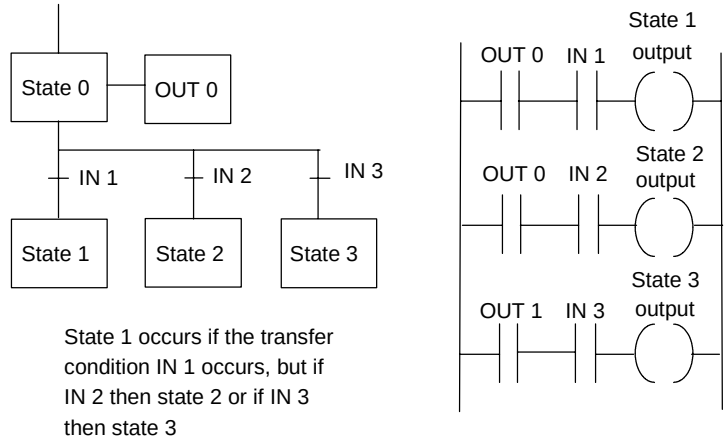


Figure 6.15 *Selective branching: the state that follows State 0 will depend on whether transition IN1, IN2 or IN3 occur.*

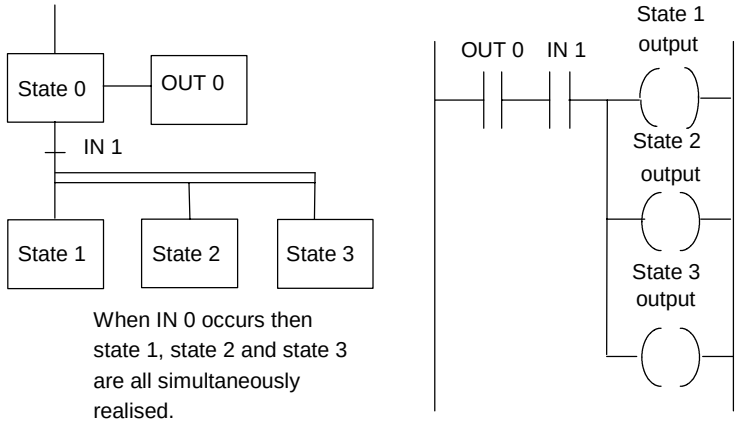


Figure 6.16 *Parallel branching states 1, 2 and 3 occur simultaneously when transition IN 1 occurs.*

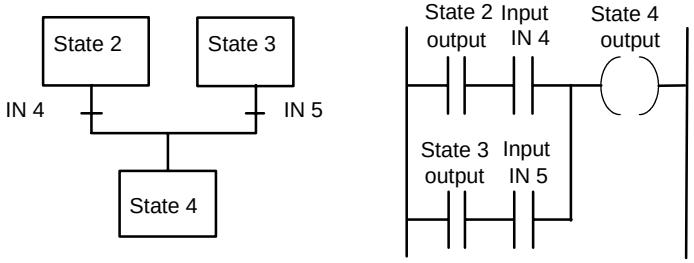


Figure 6.17 *Convergence: state 4 follows when either IN 4 or IN5 occur.*

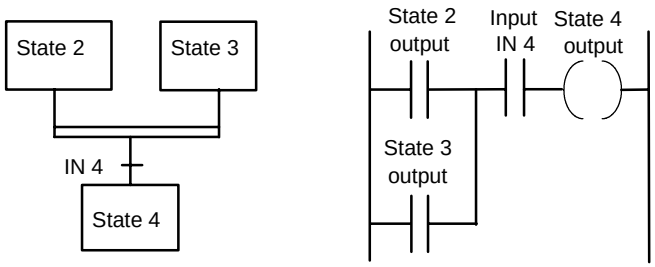


Figure 6.18 *Simultaneous convergence: when IN 4 occurs State 4 follows from either State 2 or 3.*

As an illustration of the use of the above, Figure 6.19 shows part of a program represented by both its SFC and its ladder programs.

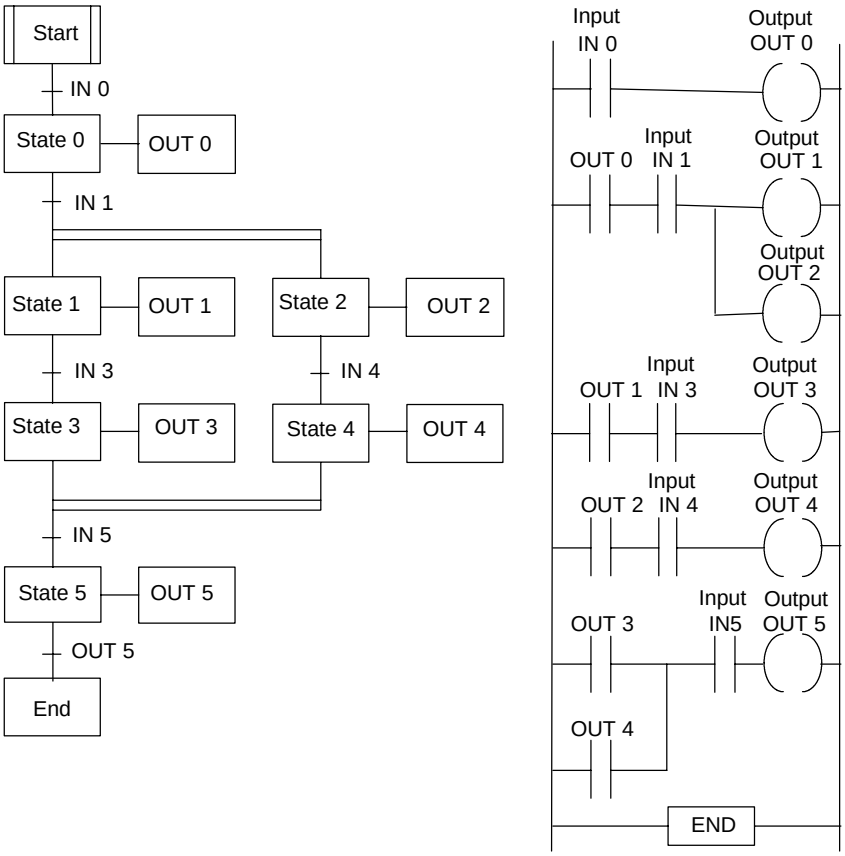


Figure 6.19 *Part of an SFC and its equivalent ladder program*

6.2.2 Actions

With states, there is an action or actions which have to be performed. Such actions, like the outputs in the above example, are depicted as rectangular boxes attached to the state. The behaviour of the action can be

given using a ladder diagram, function block diagram, instruction list or structured text. Thus, where a ladder diagram is used, the behaviour of the action is shown by the ladder diagram being enclosed within the action box. The action is then activated when there is a power flow into the action box. Figure 6.20 illustrates this.

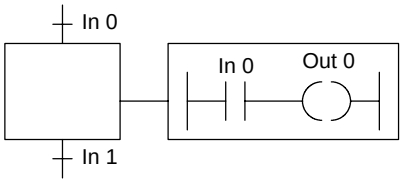


Figure 6.20 *Action represented by a ladder diagram*

Action boxes are generally preceded by qualifiers to specify the conditions to exist for the action. In the absence of a qualifier or the qualifier N, the action is not stored and executed continually while the associate state is active. The qualifier P is used for a pulse action that executes only once when a step is activated. The qualifier D is used for a time delayed action that only starts after a specified period. The qualifier L is used for a time limited action which terminates after a specified period (Figure 6.21).

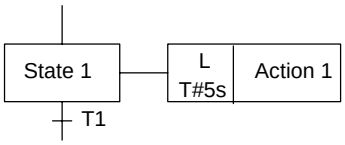


Figure 6.21 *Illustration of a qualifier used with an action, this being a time limited action*

6.3 Structured text

Structured text is a programming language that strongly resembles the programming language PASCAL. Programs are written as a series of statements separated by semicolons. The statements use predefined statements and subroutines to change variables, these being defined values, internally stored values or inputs and outputs.

Assignment statements are used to indicate how the value of a variable it to be changed, e.g.

```
Light := SwitchA;
```

is used to indicate that a light is to have its ‘value’ changed, i.e. switched on or off, when switch A changes its ‘value’, i.e. is on or off. The general format of an assignment statement is:

```
X := Y;
```

where Y represents an expression which produces a new value for the variable X. Another example is:

```
Light :- SwitchA OR SwitchB;
```

to indicate that the light is switched on by either switch A or switch B.

Table 6.2 shows some of the operators, like the OR in the above statement, that are used in structured text programs, and their relative precedence when an expression is being evaluated. For example:

```
InputA = 6
InputB = 4
InputC = 2
OutputQ := InputA/3 + InputB/(3 – InputC)
```

has (3 – InputC) evaluated before its value is used as a divisor so the second part of the OutputQ statement is 4/(3 – 2) = 4. Division has precedence over addition and so the first part of the statement is evaluated before the addition, i.e. 6/3. So we have for OutputQ the value 2 + 1 = 3.

Table 6.2 *Structured text operators*

Operator	Description	Precedence
(...)	Parenthised (bracketed) expression	Highest
Function(...)	List of parameters of a function	
**	Raising to a power	
-, NOT	Negation, Boolean NOT	
*, /, MOD	Multiplication, division, modulus operation	
+, -	Addition, subtraction	
<, >, <=, >=	Less than, greater than, less than or equal to, greater than or equal to	
=, <>	Equality, inequality	
AND, &	Boolean AND	
XOR	Boolean XOR	
OR	Boolean OR	Lowest

Structured text is not case sensitive, thus lower case or capital letters can be used as is felt necessary to aid clarity. Likewise spaces are not necessary but can be used to aid clarity, likewise indenting lines. All the identities of directly represented variables start with the % character and is followed by one or two letter code to identify whether the memory location is associated with inputs, outputs or internal memory and whether it is bits, bytes or words, e.g.

```
%IX100 (*Input memory bit 100*)
%ID200 (*Input memory word 200*)
%QX100 (*Output memory bit 100*)
```

The first letter is I for input memory location, Q for output memory location, M for internal memory. The second letter is X for bit, B for byte (8 bit), W for word (16 bits), D for double word (32 bits), L for long word (64 bits).

AT is used to fix the memory location for a variable. Thus we might have:

```
Input1 AT %IX100; (*Input1 is located at input memory bit 100*)
```

6.3.1 Conditional statements

IF ... THEN ... ELSE is used when selected statements are to be executed when certain conditions occur. For example:

```
IF (Limit_switch1 AND Workpiece_Present) THEN
Gate1 :- Open;
Gate2 :- Close;
ELSE
Gate1 :- Close;
Gate2 :- Open;
End_IF;
```

Note that the end of the IF statement has to be indicated. Another example, using PLC addresses, is:

```
IF (I:000/00 = 1) THEN
O:001/00 :- 1;
ELSE
O:000/01 = 0;
End_IF;
```

So if there is an input to I:000/00 to make it 1 then output O:001/00 is 1, otherwise it 0.

CASE is used to give the condition that selected statements are to be executed if a particular integer value occurs else some other selected statements. For example, for temperature control we might have:

```
CASE (Temperature) OF 0 ... 40 ;
Furnace_switch :- On;
40 ... 100
Furnace_switch :- Off;
ELSE
Furnace_switch :- Off;
End_CASE;
```

Note, as with all conditional statements, the end of the CASE statement has to be indicated.

6.3.2 Iteration statements

These are used where it is necessary to repeat one or more statements a number of times, depending on the state of some variable. The FOR ... DO iteration statement allows a set of statements to be repeated depending on the value of the iteration integer variable. For example:

```
FOR Input :- 10 to 0 BY -1
```

```
DO
Output :- Input;
End_FOR;
```

has the output decreasing by 1 each time the input, dropping from 10 to 0, decreases by 1.

WHILE ... DO allows one or more statements to be executed while a particular Boolean expression remains true, e.g.:

```
OutputQ :- 0;
WHILE InputA AND InputB
DO
OutputQ =: OutputQ + 1;
End_WHILE;
```

REPEAT ... UNTIL allows one or more statements to be executed and repeated whilst a particular Boolean expression remains true.

```
OutputQ :- 0
REPEAT
OutputQ = OutputQ + 1;
UNTIL (Input1 = Off) OR (OutputQ > 5)
End_REPEAT;
```

6.3.3 Structured text programs

Programs have first to define the data types required to represent data e.g.

```
TYPE Motor: ( Stopped, Running);
End_TYPE;

TYPE Valve: (Open, shut);
End_TYPE;

TYPE Pressure: REAL; (*The pressure is an analogue value*)
End_TYPE;
```

the variables, i.e. signals from sensors and output signals to be used in a program, e.g.

```
VAR_IN  (*Inputs*)
PumpFault : BOOL; (*Pump operating fault is a Boolean variable*)
End_VAR;

VAR_OUT (*Outputs*)
Motor_speed : REAL;
End_VAR;

VAR_IN
Value: INT; (*The value is an integer*)
End_VAR;
```

```
VAR
Input1 AT %IX100; (*Input1 is located at input memory bit 100*)
End_VAR;
```

and any initial values to be given to variables, e.g.

```
VAR
Temp : REAL :-100; (*Initial value is a an analogue number 100*)
End_VAR;
```

before getting down to the instruction statements.

Problems

Questions 1 to 21 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

1 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
LD      X401
AND     X402
OUT     Y430
```

describes a ladder diagram rung for which there is an output when:
(i) Input X401 is activated but X402 is not.
(ii) Input X401 and input X402 are both activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

2 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
LD      X401
OR      X402
OUT     Y430
```

describes a ladder diagram rung for which there is an output when:
(i) Input X401 is activated but X402 is not.
(ii) Input X402 is activated but X401 is not.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

3 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
LD      X401
ANI     X402
```

```
OUT     Y430
```

describes a ladder diagram rung for which there is an output when:
(i) Input X401 is activated but X402 is not.
(ii) Input X401 and input X402 are both activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

4 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
LDI     X401
ANI     X402
OUT     Y430
```

describes a ladder diagram rung for which there is an output when:
(i) Input X401 is activated but X402 is not.
(ii) Input X401 and input X402 are both activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

5 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
LD      X401
OR      Y430
ANI     X402
OUT     Y430
```

describes a ladder diagram rung for which there is:
(i) An output when input X401 is momentarily activated.
(ii) No output when X402 is activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

6 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
A       I0.1
A       I0.2
=       Q2.0
```

describes a ladder diagram rung for which there is an output when:
(i) Input I0.1 is activated but I0.2 is not.
(ii) Input I0.1 and input I0.2 are both activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

7 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
A       I0.1
O       I0.2
=       Q2.0
```

describes a ladder diagram rung for which there is an output when:
(i) Input I0.1 is activated but I0.2 is not.
(ii) Input I0.2 is activated but I0.1 is not.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

8 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
A       I0.1
AN      I0.2
=       Q2.0
```

describes a ladder diagram rung for which there is an output when:
(i) Input I0.1 is activated but I0.2 is not.
(ii) Input I0.1 and input I0.2 are both activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

9 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
AN      I0.1
AN      I0.2
=       Q2.0
```

describes a ladder diagram rung for which there is an output when:
(i) Input I0.1 is activated but I0.2 is not.
(ii) Input I0.1 and input I0.2 are both activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

10 Decide whether each of these statements is True (T) or False (F).

The instruction list:

```
A     I0.1
O     Q2.0
AN    I0.2
=     Q2.0
```

describes a ladder diagram rung for which there is:
(i) An output when input I0.1 is momentarily activated.
(ii) No output when I0.2 is activated.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

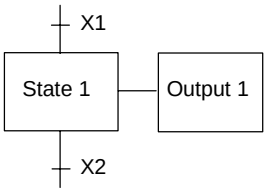


Figure 6.22 *Problem 11*

11 Decide whether each of these statements is True (T) or False (F).

For the sequential function chart shown in Figure 6.22:
(i) State 1 is realised when the condition X1 is realised.
(ii) Output 1 occurs when the condition X2 is realised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

12 Decide whether each of these statements is True (T) or False (F).

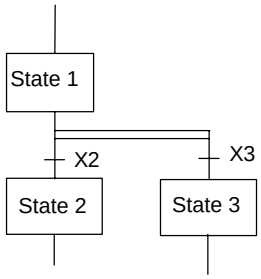


Figure 6.23 *Problem 12*

For the sequential function chart shown in Figure 6.23, if State 1 is active:
(i) State 2 is realised when the condition X2 is realised.
(ii) State 3 occurs when the condition X3 is realised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

13 For the ladder program described in Figure 6.24(a), which of the sequential function charts in Figure 6.24(b) will represent it?

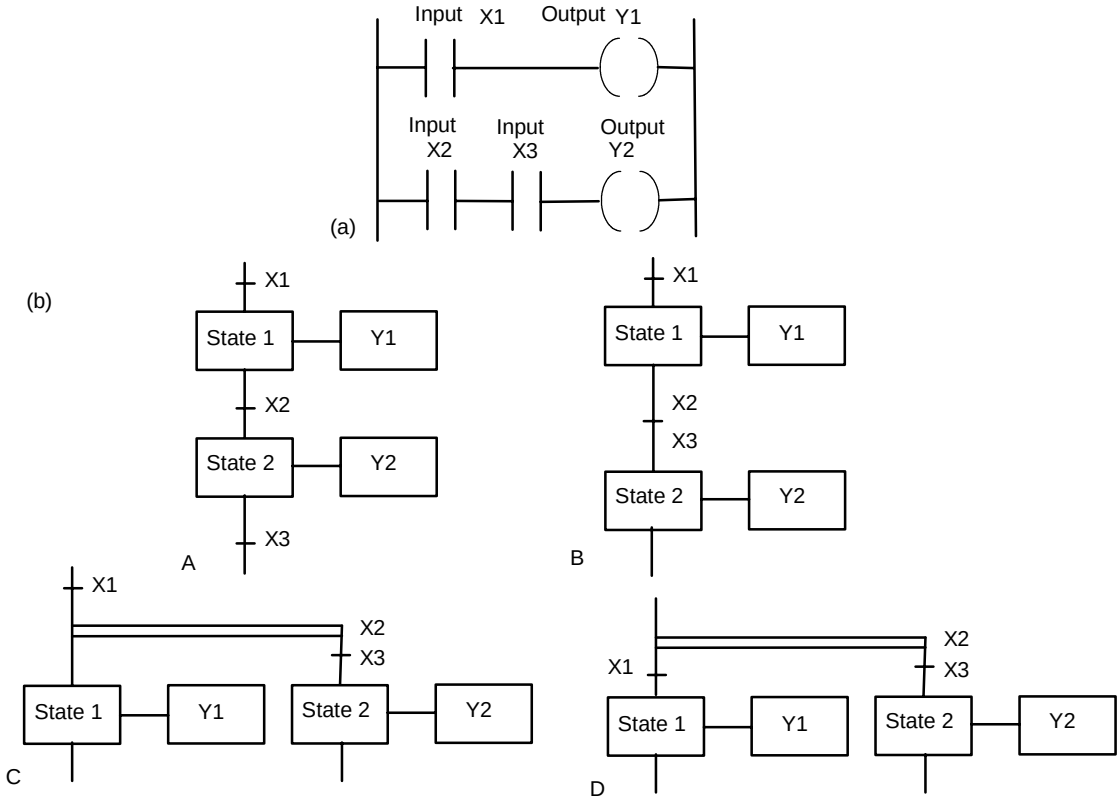


Figure 6.24 *Problem 13*

Problems 14, 15 and 16 concern the sequential function chart shown in Figure 6.25.

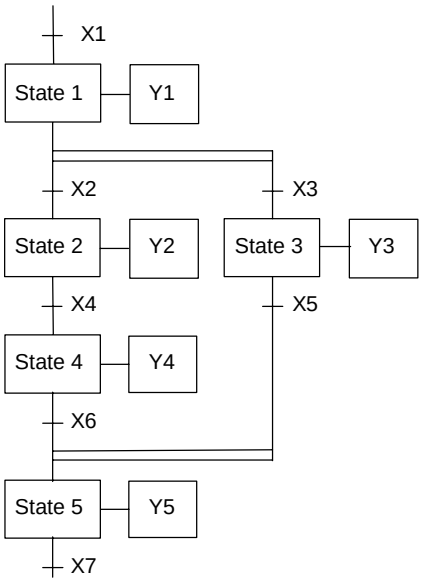


Figure 6.25 *Problems 14, 15 and 16*

14 Decide whether each of these statements is True (T) or False (F).

For the sequential function chart shown in Figure 6.25, output Y2 will occur if output Y1 has been realised and:
(i) Both X2 and X3 have been realised.
(ii) Just X2 has been realised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

15 Decide whether each of these statements is True (T) or False (F).

For the sequential function chart shown in Figure 6.25, output Y4 will occur if output Y2 has occurred and:
(i) Output Y3 has occurred.
(ii) X4 has been realised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

16 Decide whether each of these statements is True (T) or False (F).

For the sequential function chart shown in Figure 6.25, output Y5 will occur if:
(i) Output Y4 has occurred and condition X6 is realised.
(ii) Output Y3 has occurred and condition X5 is realised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

17 For the sequential function chart in Figure 6.26, which of the ladder programs in Figure 6.27 describes the same program?

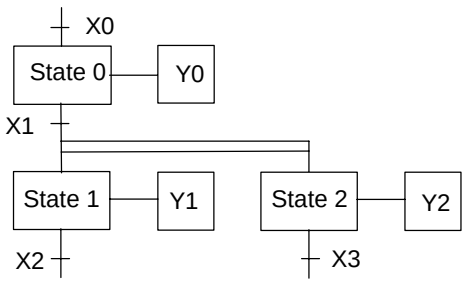


Figure 6.26 *Problem 17*

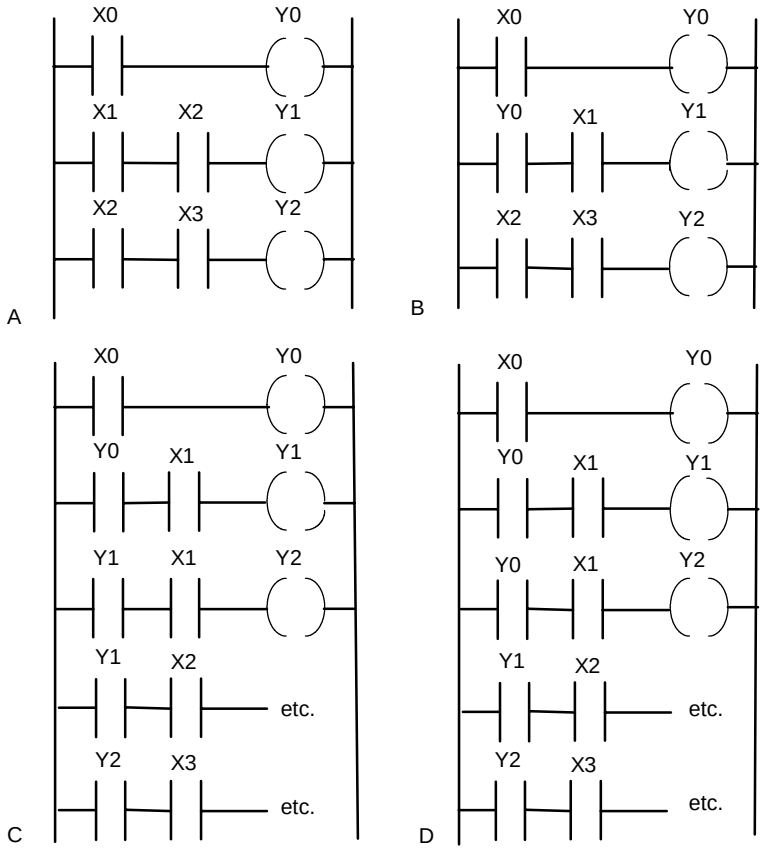


Figure 6.27 *Problem 17*

18 Decide whether each of these statements is True (T) or False (F).

For the following structured text program element:

```
VAR
  i: INT;
End_VAR;

i :- 0;
REPEAT
  i :- i + 1;
UNTIL i = 5;
End_REPEAT;
```

(i) The variable i can only have the 0 or 1 values.
(ii) Each time the program repeats, *i* has its value increased by 1.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

19 Decide whether each of these statements is True (T) or False (F).

For the following structured text program element:

```
IF Input1 THEN
Motor:- 1;
End_IF;

IF Input2 THEN
Motor:- 0;
End_IF;
```

(i) When input 1 occurs then the motor is switched on.
(ii) When input 2 occurs then the motor is switched off.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

20 Decide whether each of these statements is True (T) or False (F).

For the following structured text program element:

```
IF (Limit_switch_1 AND Workpiece_Present) Then
Gate_1 :- Open;
Gate_2 :- Closed;
ELSE
Gate_1 :- Closed;
Gate_2 :- Open;
End_IF;
```

(i) If only the workpiece is present, gate 1 is open and gate 2 closed.
(ii) If only the limit switch is activated, gate 1 is closed and gate 2 is open.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

21 Decide whether each of these statements is True (T) or False (F).

For the following structured text program element:

```
VAR
Start_Up AT %IX120;
End_VAR
Start_Up:
```

(i) Can be found at input memory location bit 120.
(ii) Has the value 120 bits.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

7 Internal relays

This chapter continues on from the previous chapters on programming and introduces *internal relays*. A variety of other terms are often used to describe these elements, e.g. *auxiliary relays*, *markers*, *flags*, *coils*, *bit storage*. These are one of the elements giving special built-in functions with PLCs and are very widely used in programming. A small PLC might have a hundred or more internal relays, some of them being battery backed so that they can be used in situations where it is necessary to ensure safe shutdown of plant in the event of power failure. Later chapters consider other common built-in elements.

7.1 Internal relays

In PLCs there are elements that are used to hold data, i.e. bits, and behave like relays, being able to be switched on or off and switch other devices on or off. Hence the term *internal relay*. Such internal relays do not exist as real-world switching devices but are merely bits in the storage memory that behave in the same way as relays. For programming, they can be treated in the same way as an external relay output and input. Thus inputs to external switches can be used to give an output from an internal relay. This then results in the internal relay contacts being used, in conjunction with other external input switches to give an output, e.g. activate a motor. Thus we might have (Figure 7.1):

On one rung of the program:
Inputs to external inputs activate the internal relay output.

On a later rung of the program:
As a consequence of the internal relay output:
internal relay contacts are activated and so control some output.

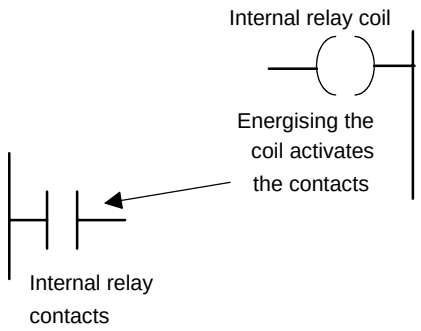


Figure 7.1 *Internal relay*

In using an internal relay, it has to be activated on one rung of a program and then its output used to operate switching contacts on another rung, or rungs, of the program. Internal relays can be programmed with as many sets of associated contacts as desired.

To distinguish internal relay outputs from external relay outputs, they are given different types of addresses. Different manufacturers tend to use different terms for internal relays and different ways of expressing their addresses. For example, Mitsubishi uses the term *auxiliary relay* or *marker* and the notation M100, M101, etc. Siemens uses the term *flag* and notation F0.0, F0.1, etc. Telemecanique uses the term *bit* and notation B0, B1, etc. Toshiba uses the term *internal relay* and notation R000, R001, etc. Allen-Bradley uses the term *bit storage* and notation in the PLC-5 of the form B3/001, B3/002, etc.

7.2 Ladder programs

With ladder programs, an internal relay output is represented using the symbol for an output device, namely ( ), with an address which indicates that it is an internal relay rather than an external relay. Thus, with a Mitsubishi PLC, we might have the address M100, the M indicating that it is an internal relay or marker rather than an external device. The internal relay switching contacts are designated with the symbol for an input device, namely | |, and given the same address as the internal relay output, e.g. M100.

7.2.1 Programs with multiple input conditions

As an illustration of the use that can be made of internal relays, consider the following situation. A system is to be activated when two different sets of input conditions are realised. We might just program this as an AND logic gate system; however, if a number of inputs have to be checked in order that each of the input conditions can be realised, it may be simpler to use an internal relay. The first input conditions then are used to give an output to an internal relay. This has associated contacts which then become part of the input conditions with the second input.

Figure 7.2 shows a ladder program for such a task. For the first rung: when input 1 or input 3 is closed and input 2 closed, then internal relay IR 1 is activated. This results in the contacts IR 1 closing. If input 4 is then activated, there is an output from output 1. Such a task might be involved in the automatic lifting of a barrier when someone approaches from either side. Input 1 and input 3 are inputs from photoelectric sensors that detect the presence of a person, approaching or leaving from either side of the barrier, input 1 being activated from one side of it and input 3 from the other. Input 2 is an enabling switch to enable the system to be closed down. Thus when input 1 or input 3, and input 2, are activated, there is an output from the internal relay 1. This will close the internal relay contacts. If input 4, perhaps a limit switch, detects that the barrier is closed then it is activated and closes. The result is then an output from Out 1, a motor which lifts the barrier. If the limit switch detects that the barrier is already open, the person having passed through it, then it opens and so output 1 is no longer energised and a counterweight might then close the barrier. The internal relay has enabled two parts of the program to be linked, one part

being the detection of the presence of a person and the second part the detection of whether the barrier is already up or down.

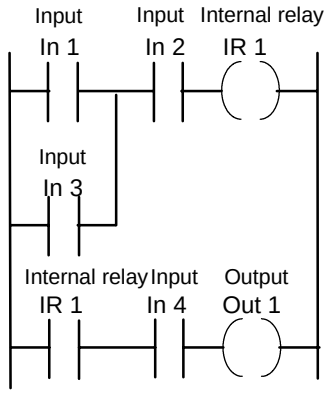


Figure 7.2 *Internal relay*

Figure 7.3(a) shows how Figure 7.2 would appear in Mitsubishi notation and Figure 7.3(b) in Siemens notation.

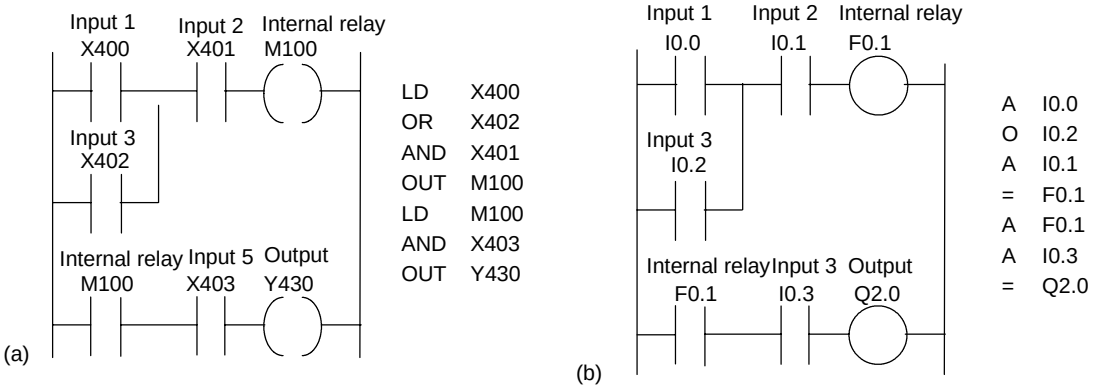


Figure 7.3 *Figure 7.2 in: (a) Mitsubishi notation, (b) Siemens notation*

Figure 7.4 is another example of this type of ladder program. The output 1 is controlled by two input arrangements. The first rung shows the internal relay IR 1 which is energised if the input In 1 or In 2 is activated and closed. The second rung shows the internal relay IR 2 which is energised if the inputs In 3 and In 4 are both energised. The third rung shows that the output Out 1 is energised if the internal relay IR 1 or IR 2 is activated. Thus there is an output from the system if either of two sets of input conditions is realised.

7.2.2 Latching programs

Another use of internal relays is for resetting a latch circuit. Figure 7.5 shows an example of such a ladder program.

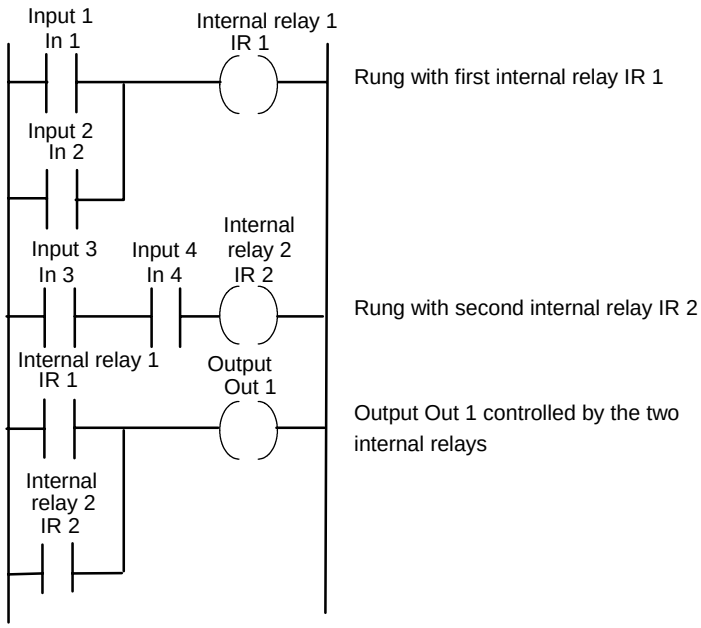


Figure 7.4 *Use of two internal relays*

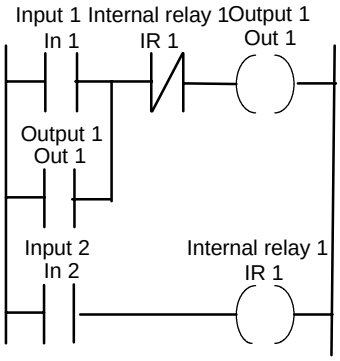


Figure 7.5 *Resetting latch*

When the input 1 contacts are momentarily closed, there is an output at Out 1. This closes the contacts for Out 1 and so maintains the output, even when input 1 opens. When input 2 is closed, the internal relay IR 1 is energised and so opens the IR 1 contacts, which are normally closed. Thus the output Out 1 is switched off and so the output is unlatched.

Consider a situation requiring latch circuits where there is an automatic machine that can be started or stopped using push-button switches. A latch circuit is used to start and stop the power being applied to the machine. The machine has several outputs which can be turned on if the power has been turned on and are off if the power is off. It would be possible to devise a ladder diagram which has individually latched

controls for each such output. However, a simpler method is to use an internal relay. Figure 7.6 shows such a ladder diagram. The first rung has the latch for keeping the internal relay IR 1 on when the start switch gives a momentary input. The second rung will then switch the power on. The third rung will also switch on and give output Out 2 if input 2 contacts are closed. The third rung will also switch on and give output Out 3 if input 3 contacts are closed. Thus all the outputs can be switched on when the start push button is activated. All the outputs will be switched off if the stop switch is opened. Thus all the outputs are latched by IR 1.

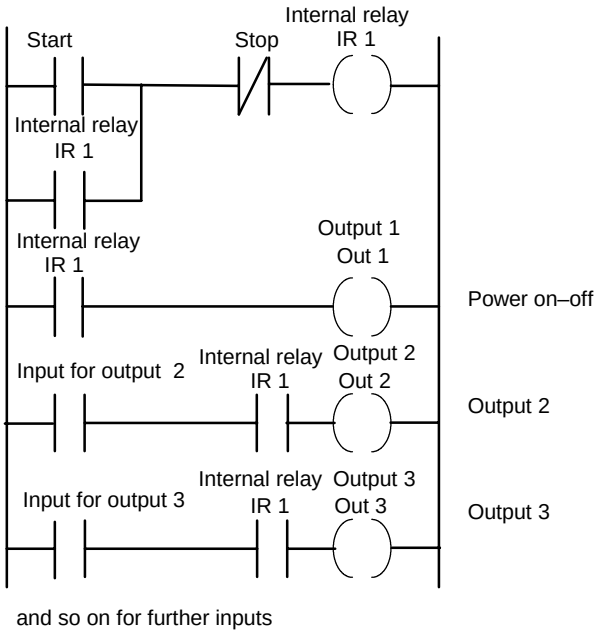


Figure 7.6 *Starting of multiple outputs*

7.3 Battery-backed relays

If the power supply is cut off from a PLC while it is being used, all the output relays and internal relays will be turned off. Thus when the power is restored, all the contacts associated with those relays will be set differently from when the power was on. Thus, if the PLC was in the middle of some sequence of control actions, it would resume at a different point in the sequence. To overcome this problem, some internal relays have battery back-up so that they can be used in circuits to ensure a safe shutdown of plant in the event of a power failure and so enable it to restart in an appropriate manner. Such battery-backed relays retain their state of activation, even when the power supply is off. The relay is said to have been made *retentive*.

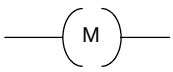


Figure 7.7 *Retentive memory coil*

The term *retentive memory coil* is frequently used for such elements. Figure 7.7 shows the IEC 1131-3 standard symbol for such elements. With Mitsubishi PLCs, battery-backed internal relay circuits use M300 to M377 as addresses for such relays. Other manufacturers use different addresses and methods of achieving retentive memory. The Allen-Bradley PLC-5 uses latch and unlatch rungs. If the relay is latched, it remains latched if

power is lost and is unlatched when the unlatch relay is activated. See Section 7.5 for a discussion of such relays in the context of set and reset coils.

As an example of the use of such a relay, Figure 7.8 shows a ladder diagram for a system designed to cope with a power failure. IR 1 is a battery-backed internal relay. When input 1 contacts close, output IR 1 is energised. This closes the IR 1 contacts, latching so that IR 1 remains on even if input 1 opens. The result is an output from Out 1. If there is a power failure, IR 1 still remains energised and so the IR 1 contacts remain closed and there is an output from Out 1.

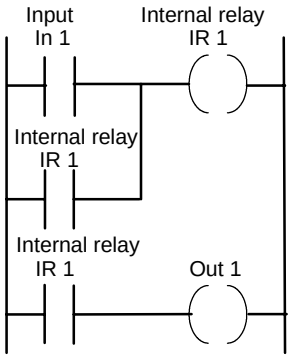


Figure 7.8 *Battery-backed relay program*

7.4 One-shot operation

One of the functions provided by some PLC manufacturers is the ability to program an internal relay so that its contacts are activated for just one cycle, i.e. one scan through the ladder program. Hence it provides a fixed duration pulse at its contacts when operated. This function is often termed *one-shot*. While some PLCs have such a function as an entity as part of their programs, such a function can easily be developed with just two rungs of a ladder program. Figure 7.9 shows such a pair of rungs. When the trigger input occurs, it gives a trigger output in rung 1. In rung 2 it gives a cycle control output on an internal relay. Because rung 2 occurs after rung 1, the effect of the cycle control is not felt until the next cycle of the PLC program when it opens the cycle control contacts in rung 1 and stops the trigger output. The trigger output then remains off, despite there being a trigger input. The trigger output can only occur again when the trigger output is switched off and then switched on again.

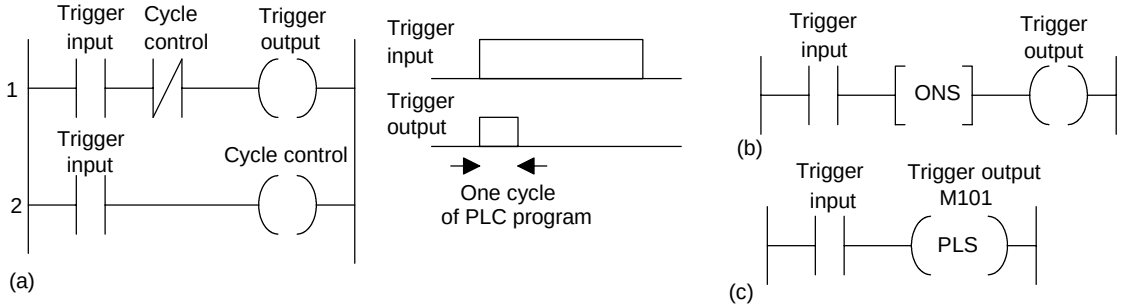


Figure 7.9 *One-shot (a) program, (b) facility in an Allen-Bradley PLC, (c) facility in a Mitsubishi PLC*

With the Mitsubishi PLC, the output internal relay, say M100, is activated when the trigger input, say X400, contacts close. Under normal circumstances, M100 would remain on for as long as the X400 contacts were closed. However, if M100 has been programmed for pulse operation, M100 only remains on for a fixed period of time, one program cycle. It then goes off, regardless of X400 being on. The programming instructions that would be used are LD X400, PLS M100. The above represents pulse operation when the input goes from off to on, i.e. is positive-going. If, in Figure 7.9(c), X400 is made normally closed, rather than normally open, then the pulse occurs when the input goes from on to off, i.e. is negative-going.

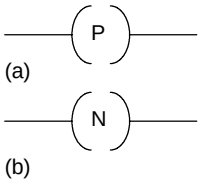
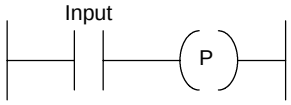


Figure 7.10 *(a) Positive transition-sensing coil, (b) negative transition-sensing coil*

The IEC 1131-3 gives standards for positive transition-sensing and negative transition-sensing coils (Figure 7.10). With the positive transition-sensing coil, if the power flow to it changes from off to on, the output is set on for one ladder rung evaluation. With the negative transition-sensing coil, if the power to it changes from off to on, the output is set on for one ladder rung evaluation. Thus, for the ladder rung of Figure 7.11, with the input off there is no output. When the input switches on, there is an output from the coil. However, the next and successive cycles of the program do not give outputs from the coil even though the switch remains on. The coil only gives an output the first time the switch is on.



Evaluation	Input	P output
1	Off	Off
2	On	On
3	On	Off
4	On	Off

Figure 7.11 *Ladder rung with a positive-transition sensing coil*

7.5 Set and reset

Another function which is often available is the ability to set and reset an internal relay. The set instruction causes the relay to self-hold, i.e. latch. It then remains in that condition until the reset instruction is received. The term *flip-flop* is often used. Figure 7.12 shows the IEC 1131-3 standards for such coils. The SET coil is switched on when power is supplied to it and remains set until it is RESET. The RESET coil is reset to the off state when power is supplied to it and remains off until it is SET.

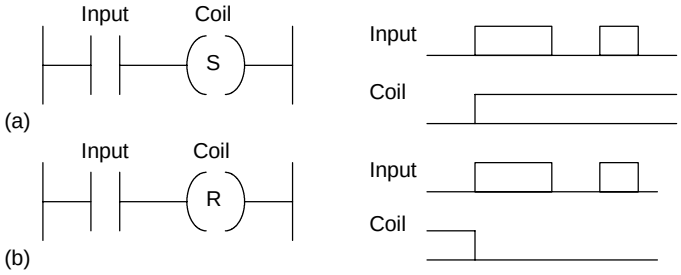


Figure 7.12 *(a) SET and (b) RESET coils*

Figure 7.13 shows an example of a ladder diagram involving such a function. Activation of the first input, X400, causes the output Y430 to be

turned on and set, i.e. latched. Thus if the first input is turned off, the output remains on. Activation of the second input, X401, causes the output Y430 to be reset, i.e. turned off and latched off. Thus the output Y430 is on for the time between X400 being momentarily switched on and X401 being momentarily switched on. Between the two rungs indicated for the set and reset operations, there could be other rungs for other activities to be carried out, the set rung switching on an output at the beginning of the sequence and off at the end.

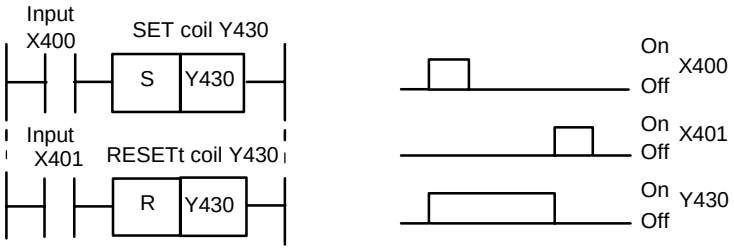


Figure 7.13 *SET and RESET*

The programming instructions for the ladder rungs in the program for Figure 7.13 are:

```
LD      X400
S       Y430
```

Other program rungs are

```
LD      X401
R       Y430
```

With a Telemecanique PLC the ladder diagram would be as shown in Figure 7.14 and the programming instructions would be:

```
L       I0,0
S       O0,0
L       I0,1
R       O0,0
```

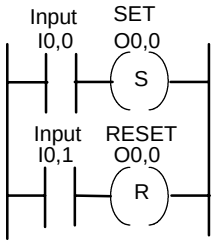


Figure 7.14 *SET and RESET*

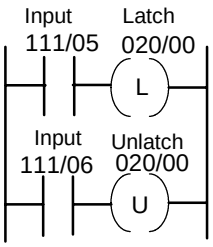


Figure 7.15 *Latch and unlatch*

With an Allen Bradley PLC, the term latch and unlatch is used. Figure 7.15 shows the ladder diagram.

The SET and RESET coil symbols are often combined into a single box symbol. Figure 7.16 shows the equivalent ladder diagram for the set-reset function in the above Figures with a Siemens PLC. The term memory box is used by them for the SET/RESET box, the box shown is termed a SR or reset priority memory function in that reset has priority. The programming instructions (F being used to indicate an internal relay) are:

```
A     I0.0
S     F0.0
A     I0.1
R     F0.0
A     F0.0
=     Q2.0
```

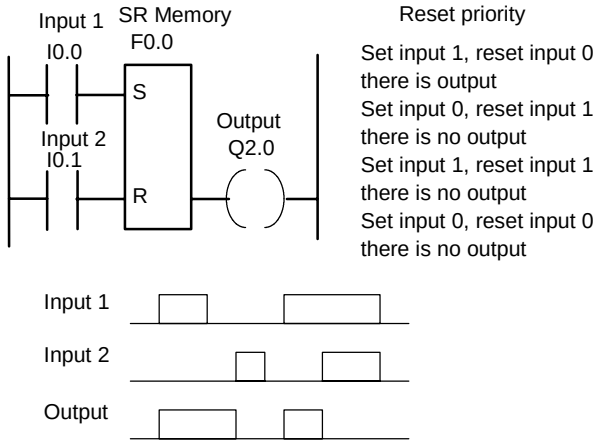


Figure 7.16 *SET and RESET, with reset priority*

With set priority (RS memory box), the arrangement is as shown in Figure 7.17.

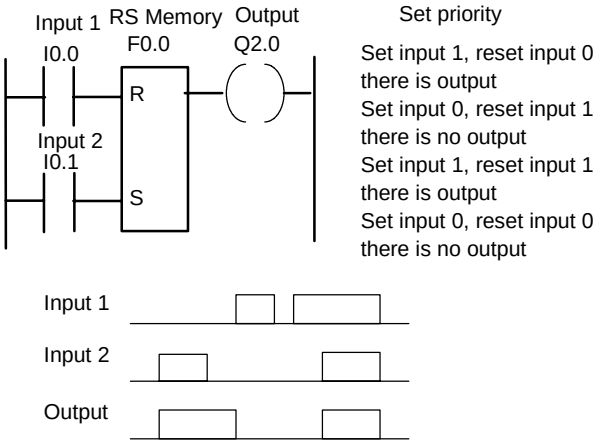


Figure 7.17 *SET and RESET, with reset priority*

Toshiba uses the term flip-flop and Figure 7.18 shows the ladder diagram.

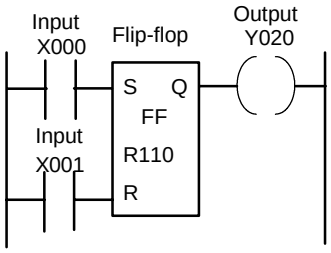


Figure 7.18 *Flip-flop*

Figure 7.19 shows how the set-reset function can be used to build the pulse (one-shot) function described in the previous section. Figure 7.19(a) shows it for a Siemens PLC (F indicates internal relay) and Figure 7.19(b) for a Telemecanique PLC (B indicates internal relay. In (a) and (b), an input (I0.0, I0,0) causes the internal relay (B0, F0.0) in the first rung to be activated. This results, second rung, in the set-reset internal relay being set. This setting action results in the internal relay (F0.1, B1) in the first rung opening and so, despite there being an input in the first rung, the internal relay (BO, F0.0) opens. However, because the rungs are scanned in sequence from top to bottom, a full cycle must elapse before the internal relay in the first rung opens. A pulse of duration one cycle has thus been produced. The system is reset when the input (I0.0, I0,0) ceases.

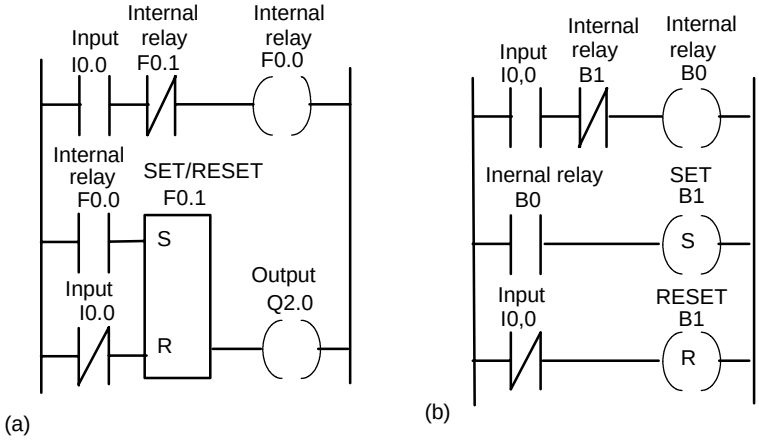


Figure 7.19 *Pulse function: (a) Siemens PLC, (b) Telemecanique PLC*

7.5.1 Program examples

An example of the basic elements of a simple program for use with a fire alarm system is shown in Figure 7.20. Fire sensors provide inputs to a SET-RESET function block so that if one of the sensors is activated the alarm is set and remains set until it is cleared by being reset. When set it sets of the alarm.

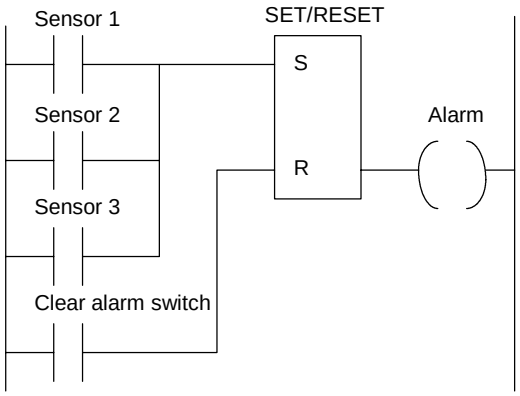


Figure 7.20 *Alarm system*

Another program showing the basic elements of a program is shown in Figure 7.21. This could be used with a system designed to detect when a workpiece has been loaded into the correct position for some further operation. When the start contacts are closed then the output causes the workpiece to move. This continues until a light beam is interrupted and resets, causing the output to cease. A stop button is available to stop the movement at any time.

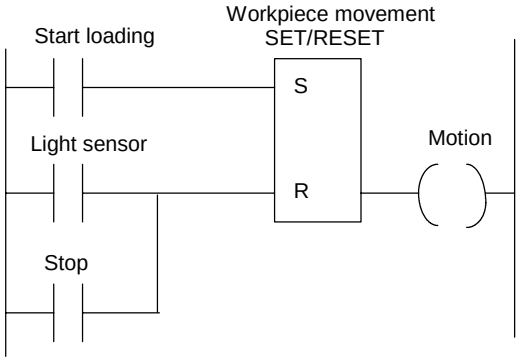


Figure 7.21 *Loading system*

7.6 Master control relay

When large numbers of outputs have to be controlled, it is sometimes necessary for whole sections of ladder diagrams to be turned on or off when certain criteria are realised. This could be achieved by including the contacts of the same internal relay in each of the rungs so that its operation affects all of them. An alternative is to use a *master control relay*.

Figure 7.22 illustrates the use of such a relay to control a section of a ladder program. With no input to input 1, the output internal relay MC 1 is not energised and so its contacts are open. This means that all the rungs between where it is designated to operate and the rung on which its reset MCR or another master control relay is located are switched off. Assuming it is designated to operate from its own rung, then we can imagine it to be located in the power line in the position shown and so rungs 2 and 3 are off. When input 1 contacts close, the master relay MC 1 is energised.

When this happens, all the rungs between it and the rung with its reset MCR 1 are switched on. Thus outputs 1 and 2 cannot be switched on by inputs 2 and 3 until the master control relay has been switched on. The master control relay 1 acts only over the region between the rung it is designated to operate from and the rung on which MCR 1 is located.

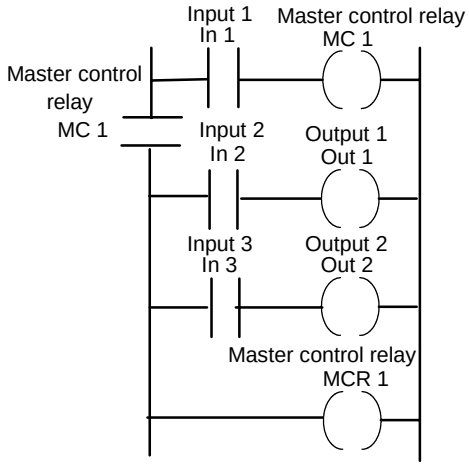


Figure 7.22 *Principle of use of a master control relay*

With a Mitsubishi PLC, an internal relay can be designated as a master control relay by programming it accordingly. Thus to program an internal relay M100 to act as master control relay contacts the program instruction is:

```
MC      M100
```

To program the resetting of that relay, the program instruction is:

```
MCR     M100
```

Thus for the ladder diagram shown in Figure 7.23, being Figure 7.22 with Mitsubishi addresses, the program instructions are:

```
LD      X400
OUT     M100
MC      M100
LD      X401
OUT     Y430
LD      X402
OUT     Y431
MC      M100
```

Figure 7.24 shows the format used by Allen-Bradley. To end the control of one master control relay (MCR), a second master control relay (MCR) is used with no contacts or logic preceding it. It is said to be programmed unconditionally.

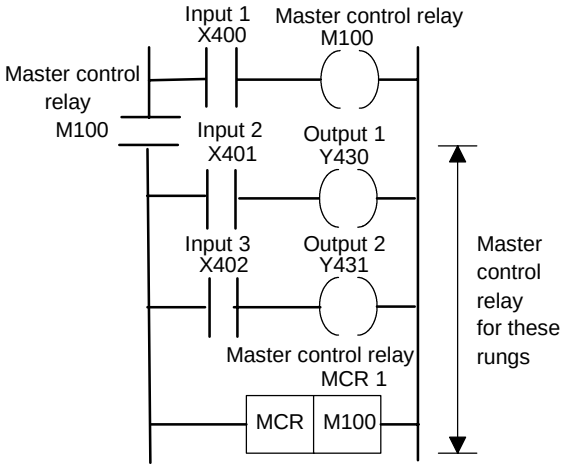


Figure 7.23 *MCR with Mitsubishi PLC*

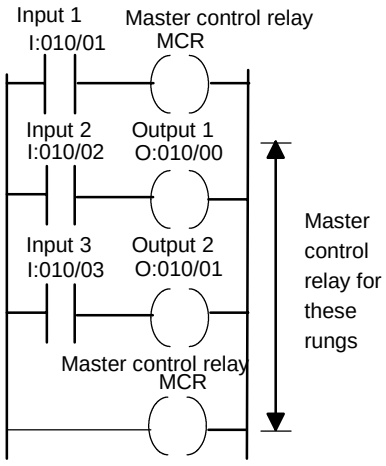


Figure 7.24 *MCR with Allen-Bradley PLC*

The representation used in Siemens ladder programs for master control relays is shown in Figure 7.25. An area in which a master control relay is to operate is defined by the activate master control area and deactivate master control relay functions. Within that area, the master control relay is enabled when MCR> coil is activated and disabled when the MCR< coil is enabled.

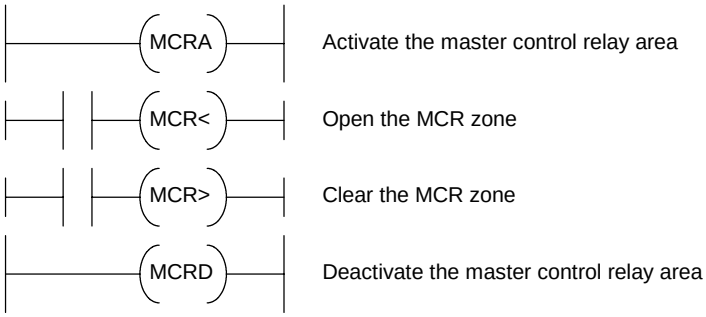


Figure 7.25 *Siemens representation of master control relays*

A program might use a number of master control relays, enabling various sections of a ladder program to be switched in or out. Figure 7.26 shows a ladder program in Mitsubishi format involving two master control relays. With M100 switched on, but M101 off, the sequence is: rungs 1, 3, 4, 6, etc. The end of the M100 controlled section is indicated by the occurrence of the other master control relay, M101. With M101 switched on, but M100 off, the sequence is: rungs 2, 4, 5, 6, etc. The end of this section is indicated by the presence of the reset. This reset has to be used since the rung is not followed immediately by another master control relay. Such an arrangement could be used to switch on one set of ladder rungs if one type of input occurs, and another set of ladder rungs if a different input occurs.

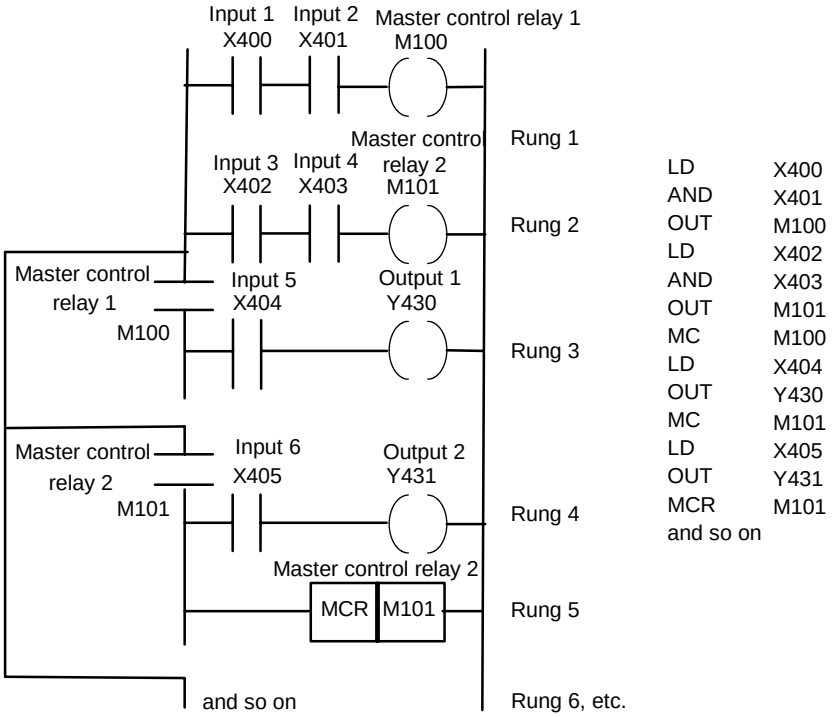


Figure 7.26 *Example showing more than one master control relay*

7.6.1 Examples of programs

The following looks at a program which illustrates the uses of master control relays. The program is being developed for use with a pneumatic valve system involving the movement of pistons in cylinders in order to give a particular sequence of piston actions. First, however, we show how latching might be used with such systems in order to maintain actions.

Consider a pneumatic system with single-solenoid controlled valves and involving two cylinders A and B, with limit switches a–, a+, b–, b+ detecting the limits of the piston rod movements (Figure 7.27), with the requirement being to give the sequence A+, B+, A–, B– . Figure 7.28 shows the ladder diagram that can be used.

The solenoid A+ is energised when the start switch is closed and limit switch b– closed. This provides latching to keep A+ energised as long as the normally closed contacts for limit switch b+ are not activated. When limit switch a+ is activated, solenoid B+ is energised. This provides latching which keeps B+ energised as long as the normally closed contacts for limit switch a– are not activated. When cylinder B extends, the limit switch b+ opens its normally closed contacts and unlatches the solenoid A+. Solenoid A thus retracts. When it has retracted and opened the normally closed contacts a–, solenoid B+ becomes unlatched and cylinder B retracts.

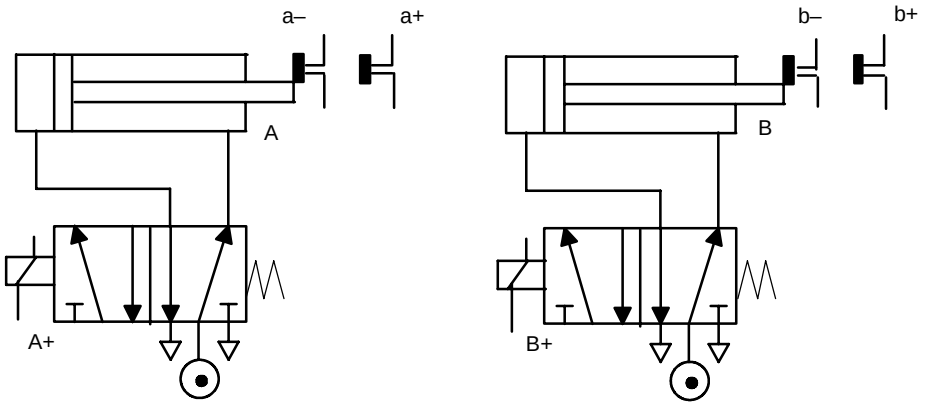


Figure 7.27 *Valve system*

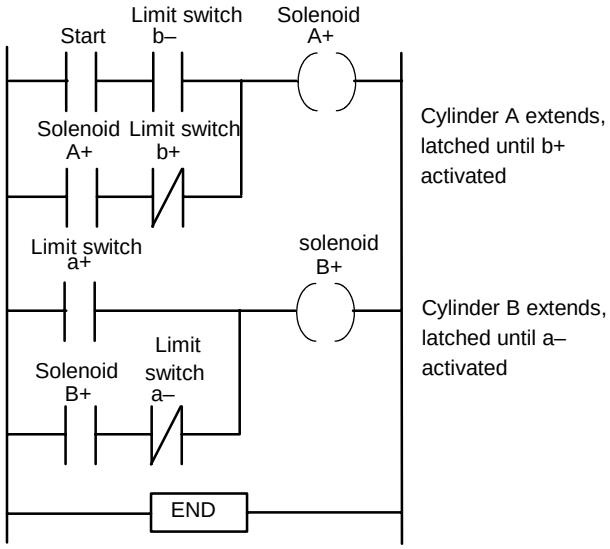


Figure 7.28 *Ladder program*

Now consider the ladder program which could be used with the pair of single-solenoid-controlled cylinders in Figure 7.27 to give, when and only when the start switch is momentarily triggered, the sequence A+, B+, A–, 10 s time delay, B– and stop at that point until the start switch is triggered again. Figure 7.29 shows how such a program can be devised using a master control relay. The master control relay is activated by the start switch and remains on until switched off by the rung containing just MCR.

Problems

Questions 1 to 21 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

Problems 1 to 3 refer to Figure 7.30 which shows a ladder diagram with an internal relay, designated IR 1, two inputs In 1 and In 2, and an output Output 1.

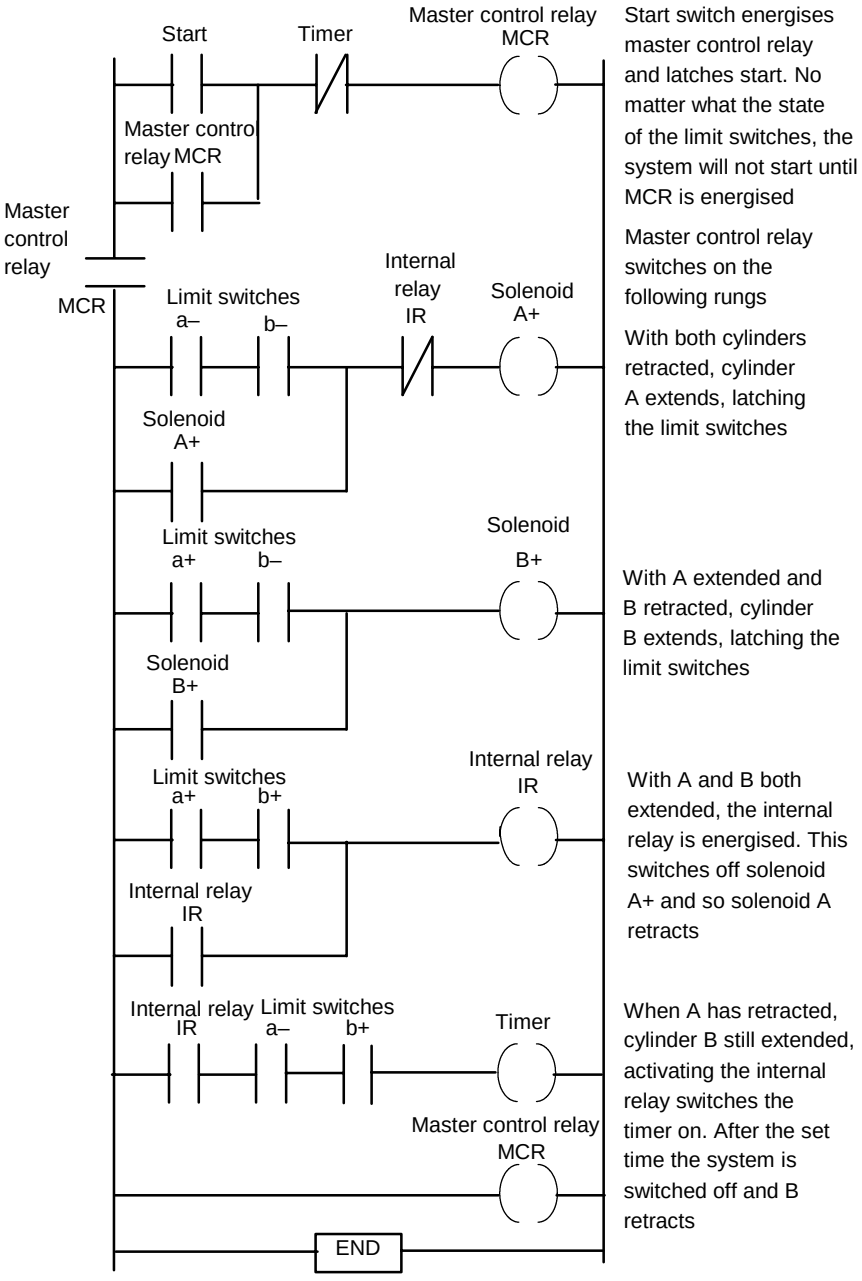


Figure 7.29 *Ladder program*

1 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 7.30, there is an output from output 1 when:

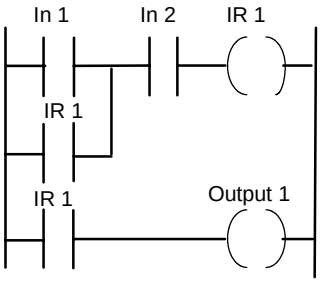


Figure 7.30 *Problems 1, 2 and 3*

(i) There is just an input to In 1.
(ii) There is just an input to In 2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

2 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 7.30, there is an output from output 1 when:
(i) There is an input to In 2 and a momentary input to In 1.
(ii) There is an input to In 1 or an input to In 2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

3 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 7.30, the internal relay:
(i) Switches on when there is just an input to In 1.
(ii) Switches on when there is an input to In 1 and to In 2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

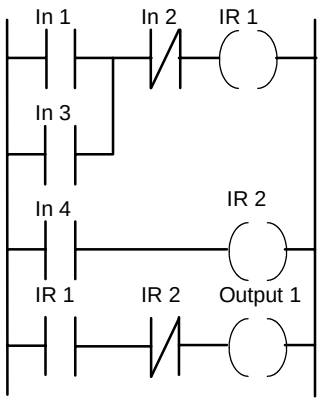


Figure 7.31 *Problems 4 to 6*

Problems 4 to 6 refer to Figure 7.31 which shows a ladder diagram involving internal relays IR 1 and IR 2, inputs In 1, In 2 and In 3, and output Output 1.

4 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 7.31, the internal relay IR 1 is energised when:
(i) There is an input to In 1.
(ii) There is an input to In 3.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

5 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 7.31, the internal relay IR 2 is energised when:
(i) Internal relay IR 1 is energised.
(ii) Input 4 is energised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

6 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 7.31, there is an output from Output 1 when:
(i) There are inputs to just In 1, In 2 and In 4.
(ii) There are inputs to just In 3 and In 4.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

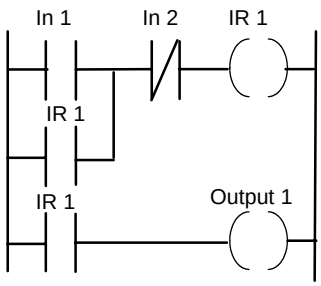


Figure 7.32 *Problems 7 and 8*

Problems 7 and 8 refer to Figure 7.32 which shows a ladder diagram involving a battery-backed relay IR 1, two inputs In 1 and In 2 and an output Output 1.

7 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 7.32, there is an output from Output 1 when:
(i) There is a short duration input to In 1.
(ii) There is no input to In 2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

8 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 7.32:
(i) The input In 1 is latched by the internal relay so that the internal relay IR 1 remains energised, even when the input In 1 ceases.
(ii) Because the internal relay IR 1 is battery-backed, once there is an output from Output 1, it will continue, even when the power is switched off, until there is an input to In 2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

9 When the program instructions LD X100, PLS M400 are used for a ladder rung, the internal relay M400 will:

A Remain on even when the input to X100 ceases.
B Remain closed unless there is a pulse input to X100.
C Remain on for one program cycle when there is an input to X100.
D Remain closed for one program cycle after an input to X100.

10 When the program instructions LDI X100, PLS M400 are used for a ladder rung, the internal relay M400 will:

A Remain on when the input to X100 ceases.
B Remain on when there is a pulse input to X100.
C Remain on for one program cycle when there is an input to X100.
D Remain on for one program cycle after the input to X100 ceases.

11 A Mitsubishi ladder program has the program instructions LD X100, S M200, LD X101, R M200, followed by other instructions for further rungs. There is the sequence: an input to the input X100, the input to X100 ceases, some time elapses, an input to the input X101, the input to X101 ceases, followed by inputs to later rungs. The internal relay M200 will remain on:

A For one program cycle from the start of the input to X100.
B From the start of the input to X100 to the start of the input to X101.
C From the start of the input to X100 to the end of the input to X101.
D From the end of the input to X100 to the end of the input to X101.

12 A Siemens ladder program has the program instructions A I0.0, S F0.0, A I0.1, R F0.0, A F0.0, = Q2.0, followed by other instructions for further rungs. There is the sequence: an input to the input I0.0, the input to I0.0 ceases, some time elapses, an input to the input I0.1, the input to I0.1 ceases, followed by inputs to later rungs. The internal relay F0.0 will remain on:

A For one program cycle from the start of the input to I0.0.
B From the start of the input to I0.0 to the start of the input to I0.1.
C From the start of the input to I0.0 to the end of the input to I0.1.
D From the end of the input to I0.0 to the end of the input to I0.1.

13 A Telemecanique ladder program has the program instructions L I0,0, S O0,0, L I0,1, R O0,0, followed by other instructions for further rungs. There is the sequence: an input to the input I0,0, the input to I0,0 ceases, some time elapses, an input to the input I0,1, the input to I0,1 ceases, followed by inputs to later rungs. The internal relay O0,0 will remain on:

A For one program cycle from the start of the input to I0,0.
B From the start of the input to I0,0 to the start of the input to I0,1.
C From the start of the input to I0,0 to the end of the input to I0,1.
D From the end of the input to I0,0 to the end of the input to I0,1.

14 An output is required from output Y430 which lasts for one cycle after an input to X100 starts. This can be given by a ladder program with the instructions:

A LD X100, Y430
B LD X100, M100, LD M100, Y 430

C LD X100, PLS M100, LD M100, Y 430
D LD X400, PLS M100, LDI M100, Y430

Questions 15 and 16 refer to Figure 7.33, which are two versions of the same ladder diagram according to two different PLC manufacturers. In (a) which uses Siemens notation, I is used for inputs, F for internal relays and Q for the output. In (b) which uses Telemecanique notation, I is used for inputs and B for internal relays.

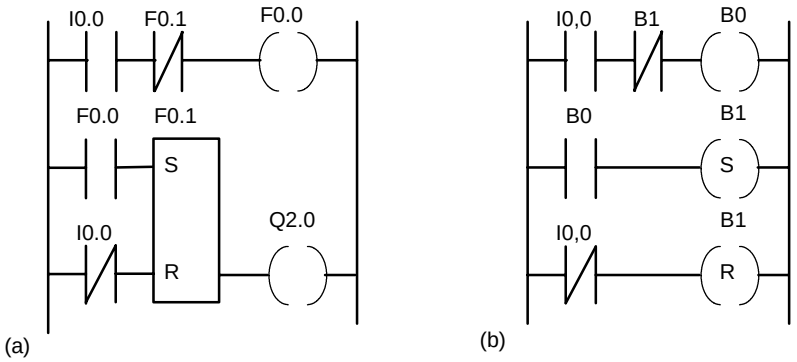


Figure 7.33 *Problems 15 and 16*

15 For the ladder diagram shown in Figure 7.33(a), when there is an input to I0.0, the output Q2.0:

A Comes on and remains on for one cycle.
B Comes on and remains on.
C Goes off and remains off for one cycle.
D Goes off and remains off.

16 For the ladder diagram shown in Figure 7.33(b), when there is an input to I0,0, the internal relay B1:

A Comes on and remains on for one cycle.
B Comes on and remains on.
C Goes off and remains off for one cycle.
D Goes off and remains off.

Problems 17 and 18 refer to Figure 7.34 which shows a Toshiba ladder program with inputs X000, X001 and X002, an output Y020 and a flip- flop R110.

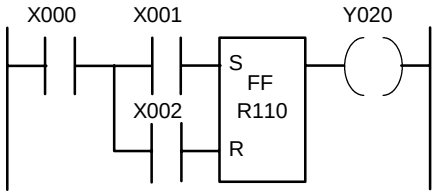


Figure 7.34 *Problems 17 and 18*

17 Decide whether each of these statements is True (T) or False (F).

For there to be an output from Y020 there must be inputs to:
(i) X000.
(ii) X001.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

18 Decide whether each of these statements is True (T) or False (F).

With an input to X000, then:
(i) An input to X001 causes the output to come on.
(ii) An input to X002 causes the output to come on.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

19 Decide whether each of these statements is True (T) or False (F).

A master control relay can be used to:
(i) Turn on a section of a program when certain criteria are met.
(ii) Turn off a section of a program when certain criteria are not met.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 20 and 21 refer to Figure 7.35, which shows a ladder program in Allen-Bradley format.

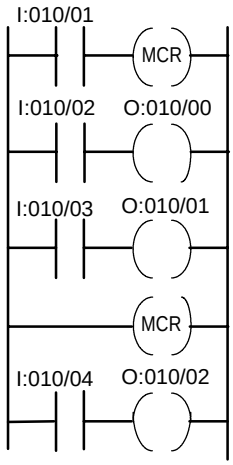


Figure 7.35 *Problems 20 and 21*

20 Decide whether each of these statements is True (T) or False (F).

When there is an input to I:010/01:
(i) An input to I:010/02 gives an output from O:010/00.
(ii) An input to I:010/03 gives an output from O:010/01.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

21 Decide whether each of these statements is True (T) or False (F).

When there is no input to I:010/01:
(i) An input to I:010/02 gives no output from O:010/00.
(ii) An input to I:010/04 gives no output from O:010/02.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

22 Devise ladder programs which can be used to:
(a) Maintain an output on, even when the input ceases and when there is a power failure.
(b) Switch on an output for a time of one cycle following a brief input.
(c) Switch on the power to a set of rungs.

8 Jump and call

This chapter considers the jump instruction which enables part of a program to be jumped over and the way in which subroutines in ladder programmes can be called up. Subroutines enable commonly occurring operations in a program to be repeatedly called up and used over again.

8.1 Jump

A function often provided with PLCs is the *conditional jump*. We can describe this as:

IF (some condition occurs) THEN
  perform some instructions
ELSE
  perform some other instructions.

Such a facility enables programs to be designed such that if certain conditions are met then certain events occur, if they are not met then other events occur. Thus, for example, we might need to design a system so that if the temperature is above 60ºC a fan is switched on, and if below that temperature no action occurs.

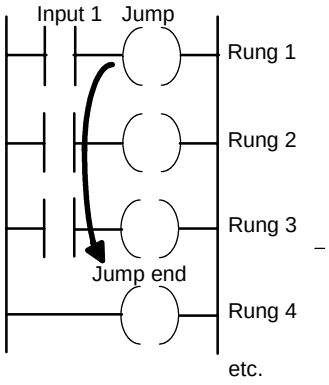


Figure 8.1 *Jump*

Thus, if the appropriate conditions are met, this function enables part of a ladder program to be jumped over. Figure 8.1 illustrates this in a general manner. When there is an input to In 1, its contacts close and there is an output to the jump relay. This then results in the program jumping to the rung in which the jump end occurs, so missing out intermediate program rungs. Thus, in this case, when there is an input to Input 1, the program jumps to rung 4 and then proceeds with rungs 5, 6, etc. When there is no input to Input 1, the jump relay is not energised and the program then proceeds to rungs 2, 3, etc.

Figure 8.2(a) shows the above ladder program in the form used by Mitsubishi. The jump instruction is denoted by CJP (conditional jump) and the place to which the jump occurs is denoted by EJP (end of jump). The condition that the jump will occur is then that there is an input to X400. When that happens the rungs involving inputs X401 and X403 are ignored and the program jumps to continue with the rungs following the end jump instruction with the same number as the start jump instruction, i.e. in this case EJP 700.

With the Allen-Bradley PLC-5 format the jump takes place from the jump instruction (JMP) to the label instruction (LBL). The JMP instruction is given a three-digit number from 000 to 255 and the LBL instruction the same number. Figure 8.2(b) shows a ladder program in this format.

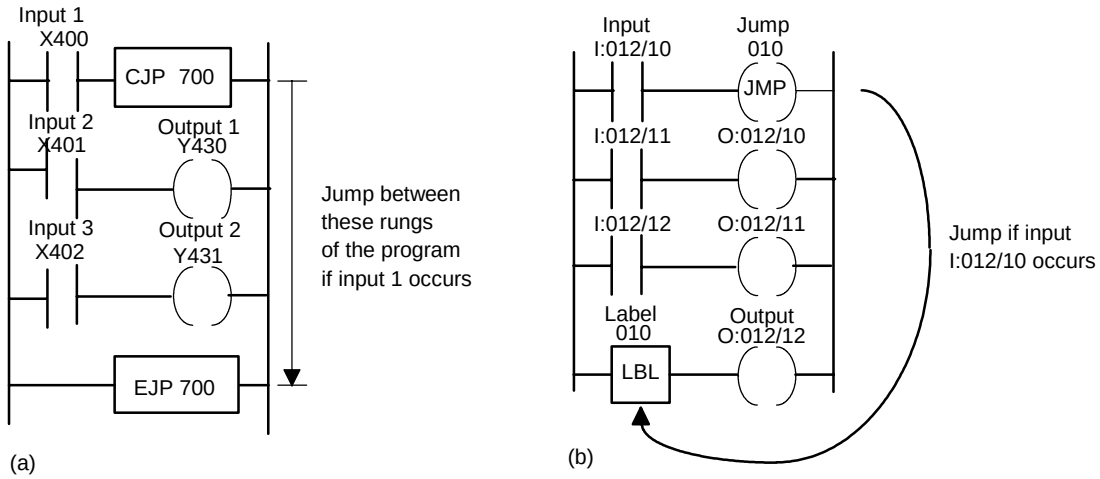


Figure 8.2 *Jump: (a) Mitsubishi program, (b) Allen-Bradley program*

With Siemens' programs, conditional jumps are represented as shown in Figure 8.3, there being a jump instruction JMP which is executed if the input is a 1 and another jump instruction JMPN which is executed if the input is 0. The end of both instructions is the label DEST.

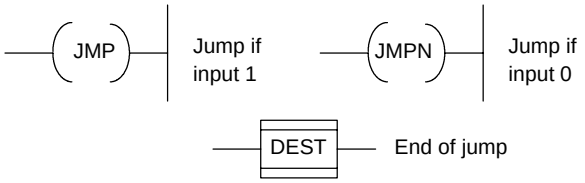


Figure 8.3 *Siemens' jump instructions*

8.1.1 Jumps within jumps

Jumps within jumps are possible. For example, we might have the situation shown in Figure 8.4. If the condition for the jump instruction 1 is realised then the program jumps to rung 8. If the condition is not met then the program continues to rung 3. If the condition for the jump instruction 2 is realised then the program jumps to rung 6. If the condition is not met then the program continues through the rungs.

Thus if we have an input to In 1, the rung sequence is rung 1, 8, etc. If we have no input to In 1 but an input to In 3, then the rung sequence is 1, 2, 6, 7, 8, etc. If we have no input to In 1 and no input to In 3, the rung sequence is 1, 2, 3, 4, 5, 6, 7, 8, etc. The jump instruction enables different groups of program rungs to be selected, depending on the conditions occurring.

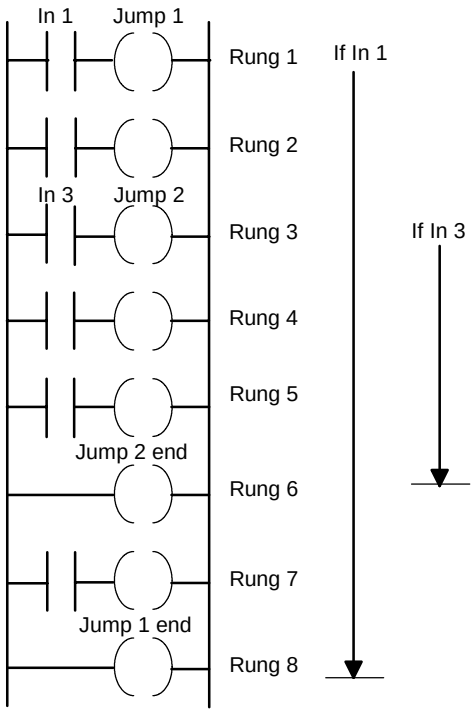


Figure 8.4 *Jumps within jumps*

8.2 Subroutines

Subroutines are small programs to perform specific tasks which can be called for use in larger programs. Thus with a Mitsubishi program we might have the situation shown in Figure 8.5(a). When input 1 occurs, the subroutine P is called. This is then executed, the instruction SRET indicating its end and the point at which the program returns to the main program. To clearly indicate where the main program ends the FEND instruction is used.

With Allen-Bradley, subroutines are called by using a jump-to-subroutine JSR instruction, the start of the subroutine being indicated by SBR and its end and point of return to the main program by RET (Figure 8.5(b)).

With Siemens a similar format can be adopted, using CALL to call up a subroutine block and RET to indicate the return instruction to the main program. However, a function box approach (Figure 8.6) can be used and is particularly useful where there is a library of subroutine functions to be called. If the EN (enable) block input is connected directly to the left power rail then the call is without conditions and always executed. If there is a logic operation preceding EN then the block call is only executed if the logic condition is fulfilled, in Figure 8.6 this is closure of contacts of Input 1. Several blocks can be connected in series by connecting the ENO, enable output, of one to the EN input of the next.

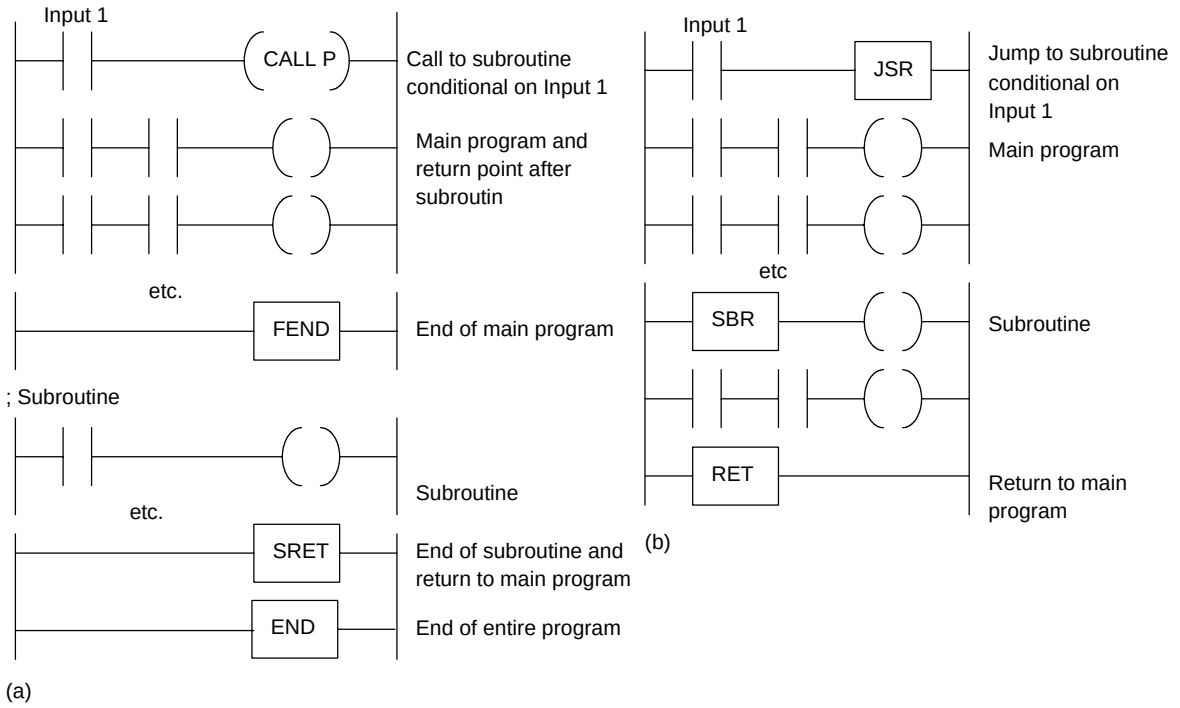


Figure 8.5 *(a) Subroutine call with Mitsubishi PLC, (b) jump to subroutine call with Allen-Bradley PLC*

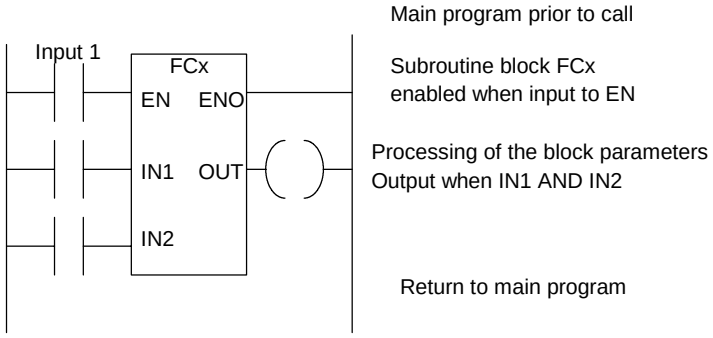


Figure 8.6 *Call to subroutine block with Siemens PLC*

Problems

Questions 1 to 4 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

Problems 1 and 2 refer to Figure 8.7 which shows a ladder diagram with inputs In 1, In 2, In 3 and In 4, outputs Out 1, Out 2, Out 3 and Out 4 and a Jump instruction.

1 For the ladder diagram shown in Figure 8.8, for output Out 1 to occur:

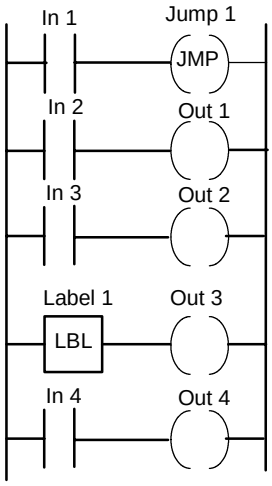


Figure 8.7 *Problems 1 and 2*

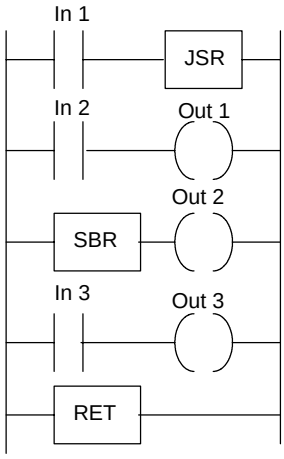


Figure 8.8 *Problems 3 and 4*

A Only input In 1 must occur
B Both inputs In 1 and In 2 must occur
C Input In 1 must not occur and input 2 must occur
D Both inputs In 1 and In 2 must not occur

2 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 8.8, following input In 1:
(i) Output 1 occurs.
(ii) Output 3 occurs.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 3 and 4 refer to Figure 8.8 which shows a ladder diagram with inputs In 1, In 2, and In 3, outputs Out 1, Out 2, Out 3 and a jump-to-subroutine instruction.

3 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 8.8:
(i) After input In 1 occurs output Out 2 occurs.
(ii) After output 3 occurs the program waits for input In 2 before proceeding

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

4 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 8.8:
(i) When input In 2 occurs, outputs 1 and 2 occur.
(ii) When input In 3 occurs, output 3 occurs.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

9 Timers

In many control tasks there is a need to control time. For example, a motor or a pump might need to be controlled to operate for a particular interval of time, or perhaps be switched on after some time interval. PLCs thus have timers as built-in devices. Timers count fractions of seconds or seconds using the internal CPU clock. This chapter shows how such timers can be programmed to carry out control tasks.

9.1 Types of timers

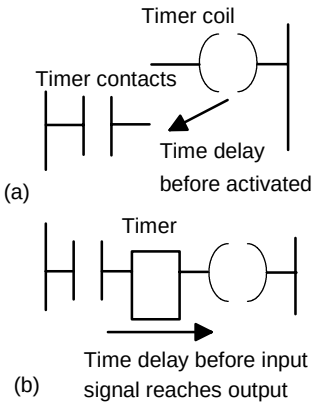


Figure 9.1 *Treatment of timers*

PLC manufacturers differ on how timers should be programmed and hence how they can be considered. A common approach is to consider timers to behave like relays with coils which when energised result in the closure or opening of contacts after some preset time. The timer is thus treated as an output for a rung with control being exercised over pairs of contacts elsewhere (Figure 9.1(a)). This is the predominant approach used in this book. Some treat a timer as a delay block which when inserted in a rung delays signals in that rung reaching the output (Figure 9.1(b)).

There are a number of different forms of timers that can be found with PLCs. With small PLCs there is likely to be just one form, the *on-delay timers*. These are timers which come on after a particular time delay (Figure 9.2(a)). *Off-delay timers* are on for a fixed period of time before turning off (Figure 9.2(b)). Another type of timer that occurs is the *pulse timer*. This timer switches on or off for a fixed period of time (Figure 9.2(c)). Figure 9.3 shows the IEC 1131-3 standard symbols for timers. TON is used to denote on-delay, TOF off-delay. TP pulse timers. On-delay is also represented by T–0 and off-delay by 0–T.

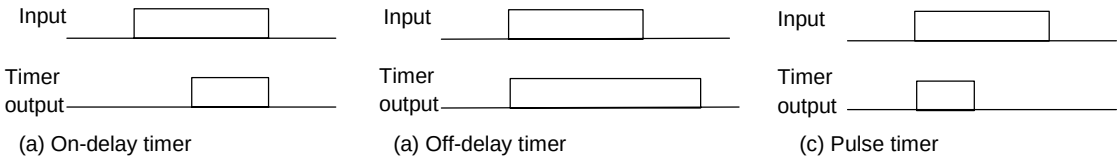


Figure 9.2 *Timers: (a) on-delay, (b) off-delay, (c) pulse*

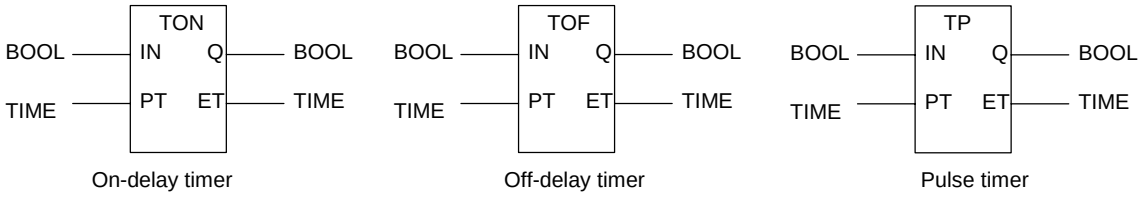


Figure 9.3 *IEC 1131-1 standards. BOOL indicates a Boolean input/output, i.e. on/off. IN is the input. Q is the output. ET is the elapsed time output. PT is the input used to specify the time.*

The time duration for which a timer has been set is termed the preset and is set in multiples of the time base used. Some time bases are typically 10 ms, 100 ms, 1 s, 10 s and 100 s. Thus a preset value of 5 with a time base of 100 ms is a time of 500 ms. For convenience, where timers are involved in this text, a time base of 1 s has been used.

9.2 Programming timers

All PLCs generally have delay-on timers, small PLCs possibly having only this type of timer. Figure 9.4(a) shows a ladder rung diagram involving a delay-on timer. Figure 9.4(a) is typical of Mitsubishi. The timer is like a relay with a coil which is energised when the input In 1 occurs (rung 1). It then closes, after some preset time delay, its contacts on rung 2. Thus the output occurs some preset time after the input In 1 occurs. Figure 9.4(b) shows the timer to be a delay item in a rung, rather than as a relay, the example being for Siemens. When the signal at the timer's start input changes from 0 to 1, the timer starts and runs for the programmed duration, giving its output then to the output coil. The time value (TV) output can be used to ascertain the amount of time remaining at any instant. A signal input of 1 at the reset input resets the timer whether it is running or not. Techniques for the entry of preset time values vary. Often it requires the entry of a constant K command followed by the time interval in multiples of the time base used. Figure 9.4(c), (d) and (e) shows ladder diagrams Telemecanique, Toshiba and Allen-Bradley.

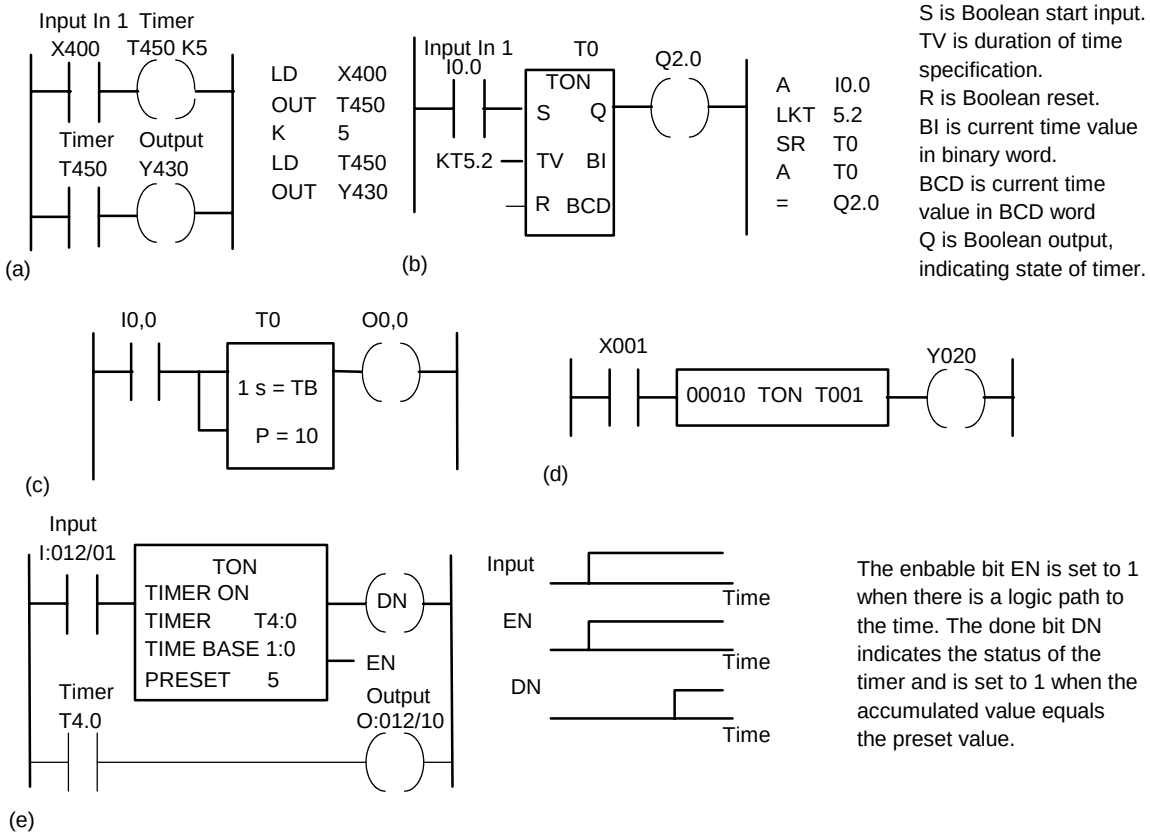


Figure 9.4 *Timers: (a) Mitsubishi, (b) Siemens, (c) Telemecanique, (d) Toshiba, (e) Allen-Bradley*

9.2.1 Sequencing

As an illustration of the use of a timer, consider the ladder diagram shown in Figure 9.5(a). When the input In 1 is on, the output Out 1 is switched on. The contacts associated with this output then start the timer. The contacts of the timer will close after the preset time delay, in this case 5.5 s. When this happens, output Out 2 is switched on. Thus, following the input In 1, Out 1 is switched on and followed 5.5 s later by Out 2. This illustrates how timed sequence of outputs can be achieved. Figure 9.5(b) shows the same operation where the format used by the PLC manufacturer is for the timer to institute a signal delay.

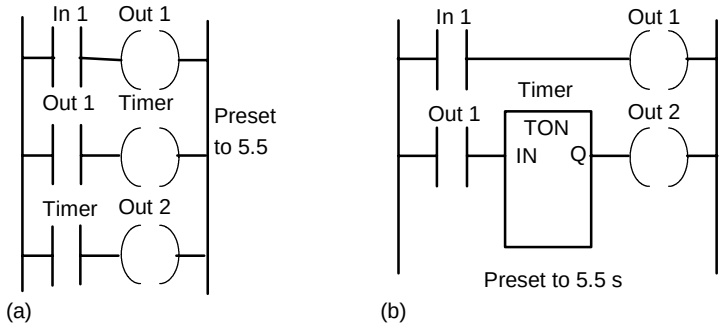


Figure 9.5 *Sequenced outputs*

Figure 9.6 shows two versions of how timers can be used to start three outputs, e.g. three motors, in sequence following a single start button being pressed. In (a) the timers are programmed as coils, whereas in (b) they are programmed as delays. When the start push button is pressed there is an output from internal relay IR1. This latches the start input. It also starts both the timers, T1 and T2, and motor 1. When the preset time for timer 1 has elapsed then its contacts close and motor 2 starts. When the preset time for timer 2 has elapsed then its contacts close and motor 3 starts. The three motors are all stopped by pressing the stop push button. Since this is seen as a complete program, the end instruction has been used.

9.2.2 Cascaded timers

Timers can be linked together, the term *cascaded* is used, to give longer delay times than are possible with just one timer. Figure 9.7(a) shows the ladder diagram for such an arrangement. Thus we might have timer 1 with a delay time of 999 s. This timer is started when there is an input to In 1. When the 999 s time is up, the contacts for timer 1 close. This then starts timer 2. This has a delay of 100 s. When this time is up, the timer 2 contacts close and there is an output from Out 1. Thus the output occurs 1099 s after the input to In 1. Figure 9.7(b) shows the Mitsubishi version of this ladder diagram and the program instructions for that ladder.

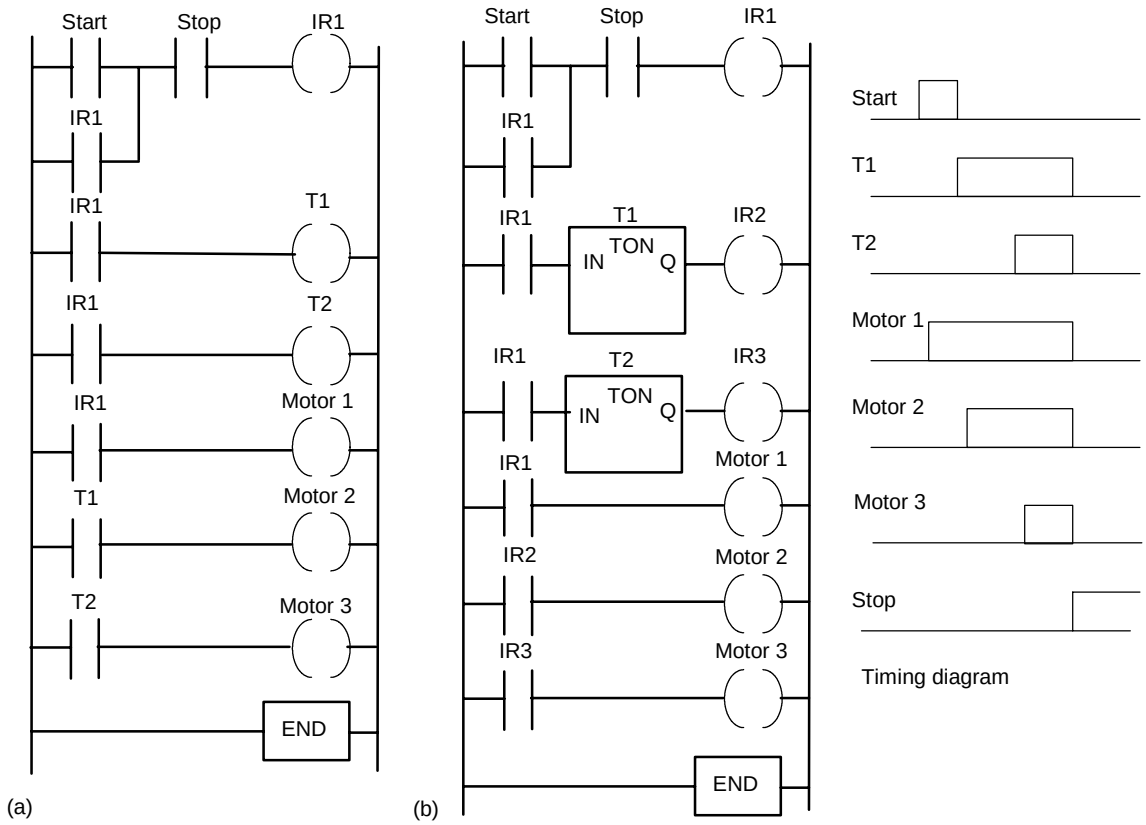


Figure 9.6 *Motor sequence*

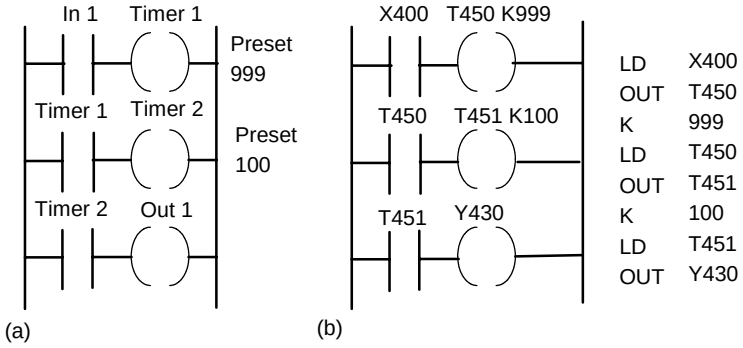


Figure 9.7 *Cascaded timers*

9.2.3 On-off cycle timer

Figure 9.8 shows how on-delay timers can be used to produce an *on-off cycle timer*. The timer is designed to switch on an output for 5 s, then off for 5 s, then on for 5 s, then off for 5 s, and so on. When there is an input to In 1 and its contacts close, timer 1 starts. Timer 1 is set for a delay of 5 s. After 5 s, it switches on timer 2 and the output Out 1. Timer 2 has a delay of 5 s. After 5 s, the contacts for timer 2, which are normally closed,

open. This results in timer 1, in the first rung, being switched off. This then causes its contacts in the second rung to open and switch off timer 2. This results in the timer 2 contacts resuming their normally closed state and so the input to In 1 causes the cycle to start all over again.

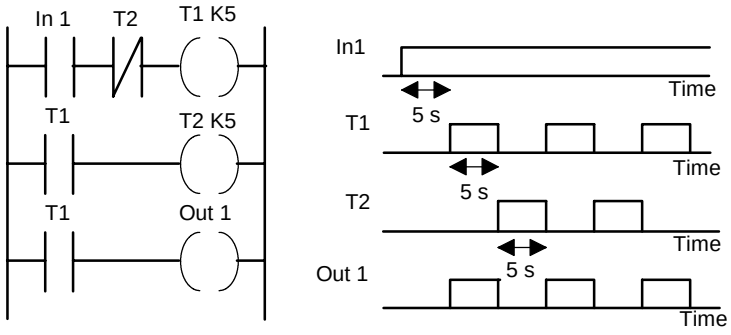


Figure 9.8 *On - off cycle timer*

Figure 9.9 shows how the above ladder program would appear in the format used with a timer considered as a delay, rather than as a coil. This might, for example, be with Siemens or Toshiba. When input In 1 closes, the timer T1 starts. After its preset time, there is an output to Out 1 and timer T2 starts. After its preset time there is an output to the internal relay IR1. This opens its contacts and stops the output from Out 1. This then switches off timer T2. The entire cycle can then repeat itself.

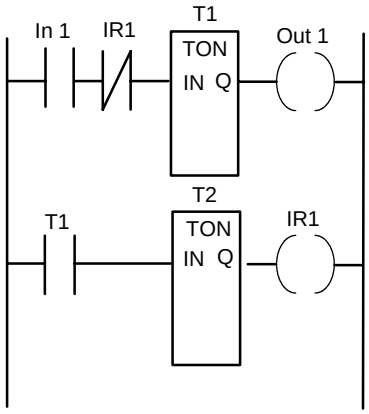


Figure 9.9 *On–off cycle timer*

9.3 Off-delay timers

Figure 9.10 shows how a on-delay timer can be used to produce an *off-delay timer*. With such an arrangement, when there is a momentary input to In 1, both the output Out 1 and the timer are switched on. Because the input is latched by the Out 1 contacts, the output remains on. After the preset timer time delay, the timer contacts, which are normally closed, open and switch off the output. Thus the output starts as on and remains on until the time delay has elapsed.

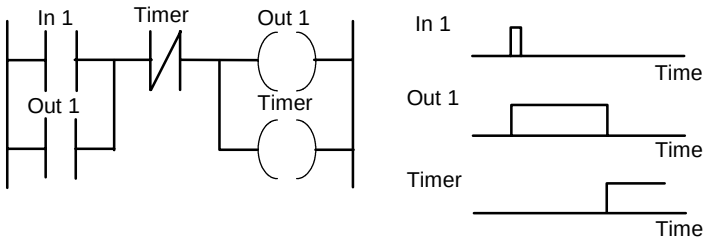


Figure 9.10 *Off-delay timer*

Some PLCs have, as well as on-delay timers, built-in off-delay timers and thus there is no need to use an on-delay timer to produce an off-delay timer. Figure 9.11 illustrates this for a Siemens PLC, giving the ladder diagram and the instruction list. Note that with this manufacturer, the timer is considered to be a delay item in a rung, rather than as a relay. In the rectangle symbol used for the timer, the 0 precedes the T and indicates that it is an on-delay timer.

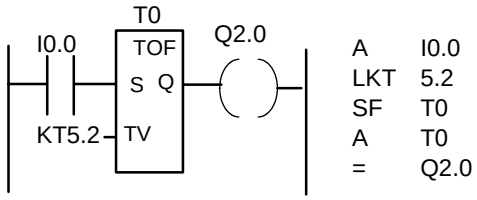


Figure 9.11 *Off-delay timer*

As an illustration of the use of an off-delay timer, consider the Allen-Bradley program shown in Figure 9.12. TOF is used to indicate that it is an off-delay, rather than on-delay (TON) timer. The time base is set to 1:0 which is 1 s. The preset is 10 so the timer is preset to 10 s.

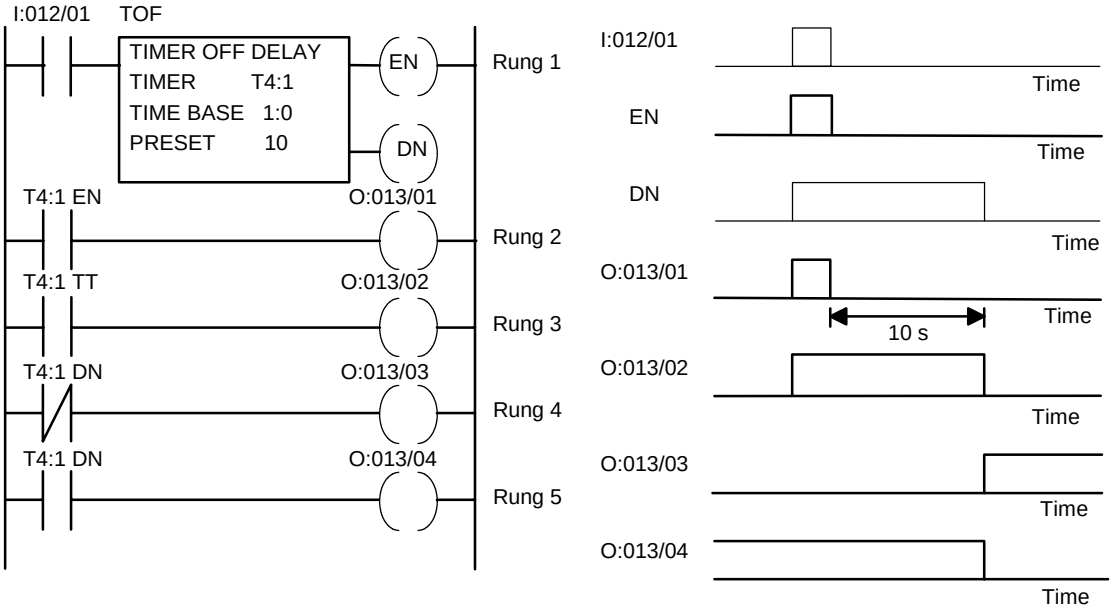


Figure 9.12 *Application of an off-delay timer*

In the first rung, the output of the timer is taken from the EN (for enable) contacts. This means that there is no time delay between an input to I:012/01 and the EN output. As a result the EN contacts in rung 2 close immediately there is an I:012/01 input. Thus there is an output from O:013/01 immediately the input I:012/01 occurs. The TT (for timer timing) contacts in rung 3 are energised just while the timer is running. Because the timer is an off-delay timer, the timer is turned on for 10 s before turning off. Thus the TT contacts will close when the set time of 10 s is running. Hence output O:012/02 is switched on for this time of 10 s. The DN (for done) contacts which are normally closed, open after the 10 s and so output O:013/03 comes on after 10 s. The DN contacts which are normally open, close after 10 s and so output O:013/04 goes off after 10 s.

9.4 Pulse timers

Pulse timers are used to produce a fixed duration output from some initiating input. Figure 9.13(a) shows a ladder diagram for a system that will give an output from Out 1 for a predetermined fixed length of time when there is an input to In 1, the timer being one involving a coil. There are two outputs for the input In 1. When there is an input to In 1, there is an output from Out 1 and the timer starts. When the predetermined time has elapsed, the timer contacts open. This switches off the output. Thus the output remains on for just the time specified by the timer.

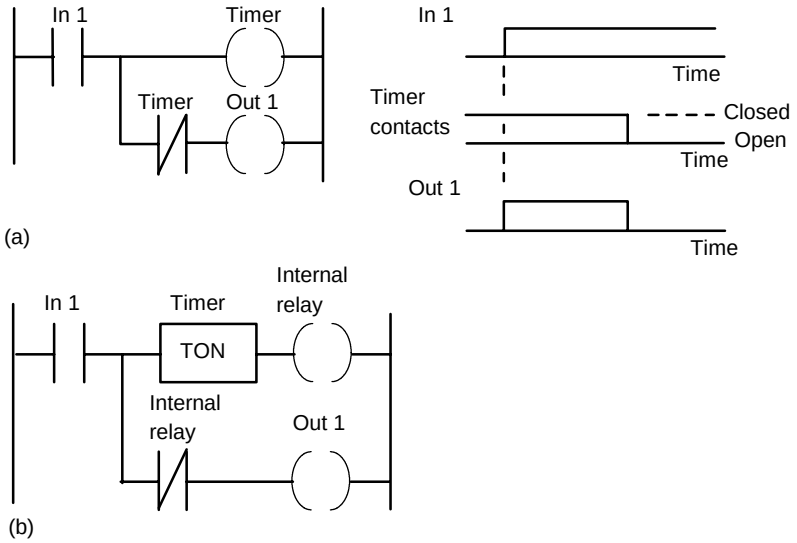


Figure 9.13 *Pulse-on timer*

Figure 9.13(b) shows an equivalent ladder diagram to Figure 9.13(a) but employing a timer which produces a delay in the time taken for a signal to reach the output.

In Figure 9.13, the pulse timer has an output switched on by an input for a predetermined time, then switching off. Figure 9.14 shows another pulse timer that switches an output on for a predetermined time after the input ceases. This uses a timer and two internal relays. When there is an

input to In 1, the internal relay IR 1 is energised. The timer does not start at this point because the normally closed In 1 contacts are open. The closing of the IR 1 contacts means that the internal relay IR 2 is energised. There is, however, no output from Out 1 at this stage because, for the bottom rung, we have In 1 contacts open. When the input to In 1 ceases, both the internal relays remain energised and the timer is started. After the set time, the timer contacts, which are normally closed, open and switch off IR 2. This in turn switches off IR 1. It also, in the bottom rung, switches off the output Out 1. Thus the output is off for the duration of the input, then being switched on for a predetermined length of time.

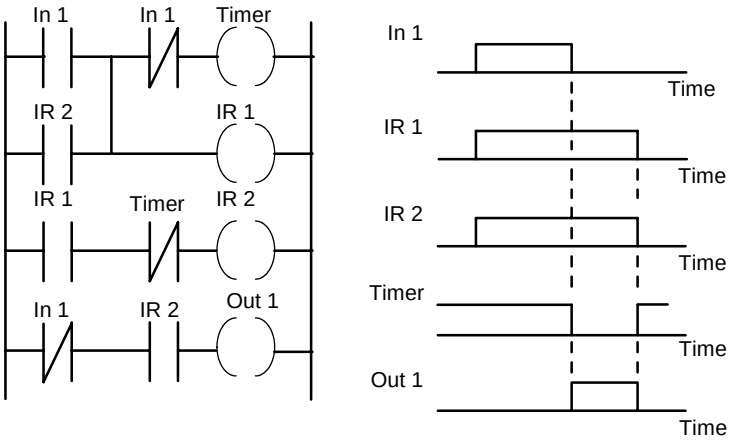


Figure 9.14 *Pulse timer on, when output ceases*

9.5 Programming examples

Consider a program (Figure 9.15) that could be used to flash a light on and off as long as there is some output occurring. Thus we might have both timer 0 and timer 1 set to 1 s. When the output occurs, then timer 0 starts and switches on after 1 s. This closes the timer 0 contacts and starts timer 1. This switches on after 1 s and, in doing so, switches off timer 0. In so doing, it switches off itself. The lamp is only on when timer 0 is on and so we have a program to flash the lamp on and off as long as there is an output.

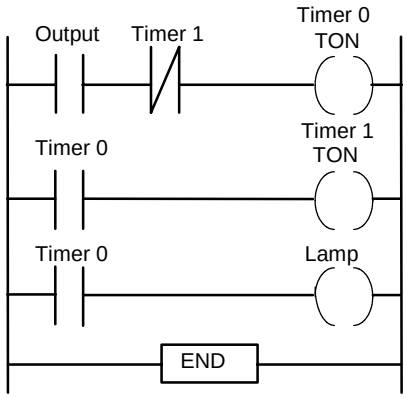


Figure 9.15 *Flashing light*

As an illustration of programming involving timers consider the sequencing of traffic lights to give the sequence red only, red plus amber, green, amber, then repeat itself. A simple system might just have the sequence triggered by time, with each of the possible states occurring in sequence for a fixed amount of time. Figure 9.16 shows the sequential function chart and a possible ladder program to give the sequence.

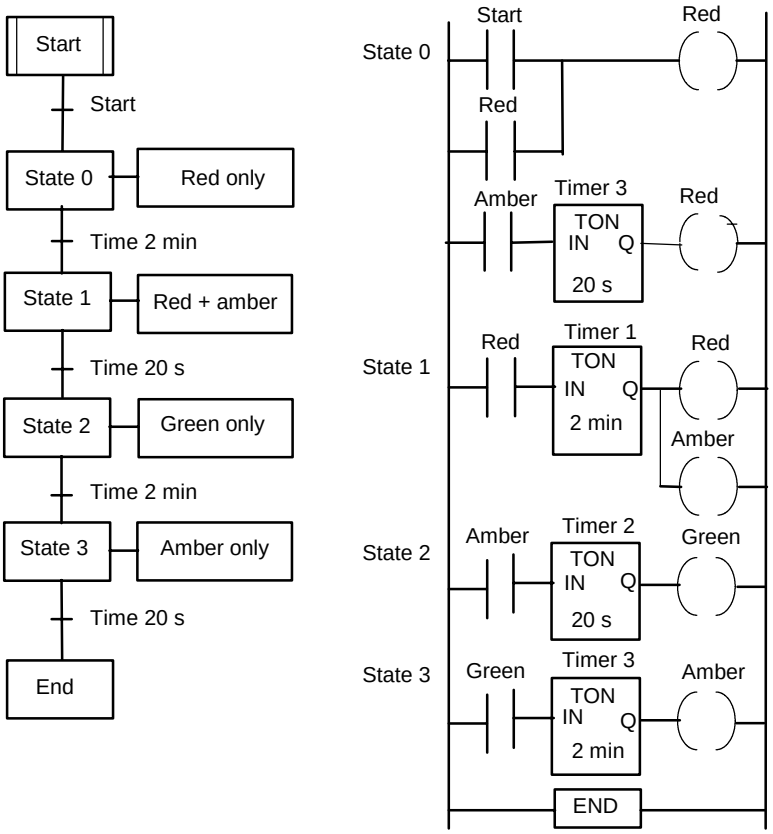


Figure 9.16 *Traffic light sequence*

Problems

Questions 1 to 19 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

Problems 1 to 3 refer to Figure 9.17 which shows a ladder diagram with an on-delay timer, an input In 1 and an output Out 1.

1 Decide whether each of these statements is True (T) or False (F).

When there is an input to In 1 in Figure 9.17:
(i) The timer starts.
(ii) There is an output from Out 1.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

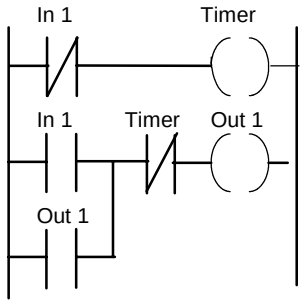


Figure 9.17 *Problems 1 to 3*

2 Decide whether each of these statements is True (T) or False (F).

The timer in Figure 9.17 starts when:
(i) There is an output.
(ii) The input ceases.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

3 Decide whether each of these statements is True (T) or False (F).

When there is an input to In 1, the output is switched:
(i) On for the time for which the timer was preset.
(ii) Off for the time for which the timer was preset.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 4 to 6 refer to Figure 9.18 which shows two alternative versions of a ladder diagram with two inputs In 1 and In 2, two outputs Out 1 and Out 2 and an on-delay timer.

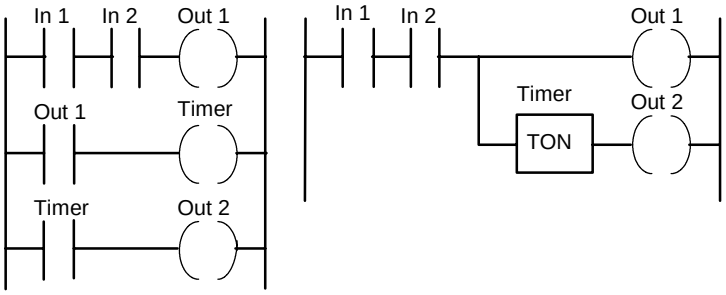


Figure 9.18 *Problems 4 to 6*

4 Decide whether each of these statements is True (T) or False (F).

When there is just an input to In 1:
(i) The timer starts.
(ii) There is an output from Out 2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

5 Decide whether each of these statements is True (T) or False (F).

When there is just an input to In 2:
(i) The timer starts.
(ii) There is an output from Out 2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

6 Decide whether each of these statements is True (T) or False (F).

When there is an input to In 1 and no input to In 2, there is an output from Out 2 which:
(i) Starts immediately.
(ii) Ceases after the timer preset time.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

7 The program instruction list for a Mitsubishi PLC is: LD X400, OUT T450, K 6, LD T450, OUT Y430. An input to X400 gives:

A An output which is on for 6 s then off for 6 s.
B An output which lasts for 6 s.
C An output which starts after 6 s.
D An output which is off for 6 s, then on for 6 s.

8 The program instruction list for a Telemecanique PLC is: L I0,0, = T0, L T0, = O0,0. When there is an input to I0,0 there is:

A An output which is on for 6 s then off for 6 s.
B An output which lasts for 6 s.
C An output which starts after 6 s.
D An output which is off for 6 s, then on for 6 s.

Problems 9 and 10 refer to the program instruction list for a Mitsubishi PLC: LD X400, OR Y430, ANI T450, OUT Y430, LD X401, OR M100, AND Y430, OUT T450, K 10, OUT M100.

9 Decide whether each of these statements is True (T) or False (F).

When there is a momentary input to X400:
(i) The output from Y430 starts.
(ii) The output from Y430 ceases after 10 s.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

10 Decide whether each of these statements is True (T) or False (F).

The output from Y430:
(i) Starts when there is a momentary input to X401.
(ii) Ceases 10 s after the input to X401.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 11 and 12 refer to Figure 9.19 which shows a system with an input In 1, an on-delay timer and an output Out 1. The timer is set for a time of 6 s. The graph shows how the signal to the input varies with time.

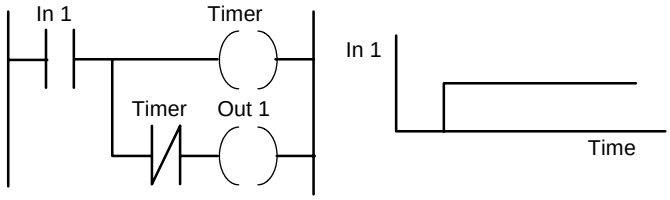


Figure 9.19 *Problems 11 and 12*

11 Decide whether each of these statements is True (T) or False (F).

The output from Out 1:
(i) Starts when the input starts.
(ii) Ceases 6 s after the start of the input.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

12 Decide whether each of these statements is True (T) or False (F).

The timer contacts:
(i) Remain closed for 6 s after the start of the input.
(ii) Open 6 s after the input starts.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 13 to 15 refer to Figure 9.20 which shows a ladder program for a Toshiba PLC involving internal relays, denoted by the letter R, and a TON timer with a preset of 20 s.

13 Decide whether each of these statements is True (T) or False (F).

The internal relay R000 in Figure 9.20:
(i) Is used to latch the input X001.
(ii) Is used to start the timer T001.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

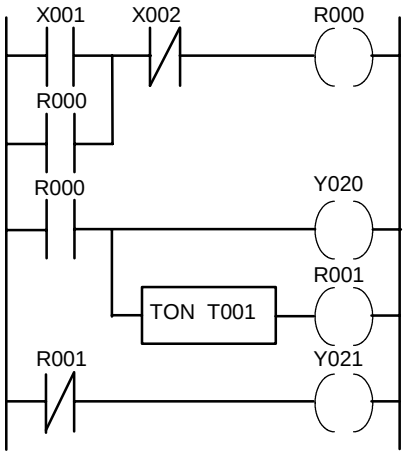


Figure 9.20 *Problems 13 to 15*

14 Decide whether each of these statements is True (T) or False (F).

With no input to X002 in Figure 9.20, the output Y020 is:
(i) Energised when there is an input to X001.
(ii) Ceases when there is no input to X001.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

15 Decide whether each of these statements is True (T) or False (F).

With no input to X002 in Figure 9.20:
(i) The output Y021 is switched on 20 s after the input X001.
(ii) The output Y020 is switched off 20 s after the input X001.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 16 to 19 refer to Figure 9.21 which shows an Allen-Bradley program and Figure 9.22 which shows a number of time charts for a particular signal input to I:012/01.

16 For the input shown in Figure 9.22, which is the output from O:013/01?

17 For the input shown in Figure 9.22, which is the output from O:013/02?

18 For the input shown in Figure 9.22, which is the output from O:013/03?

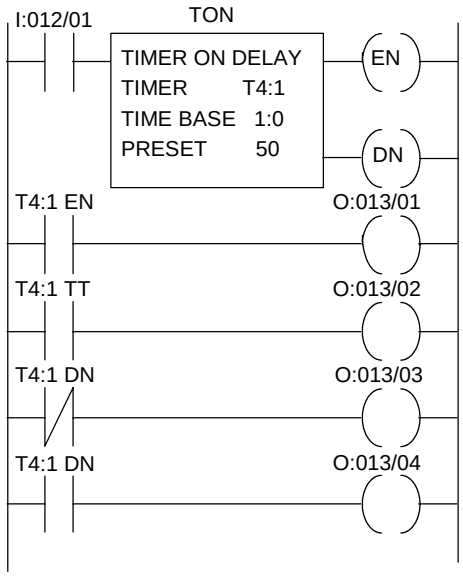


Figure 9.21 *Problems 16 to 19*

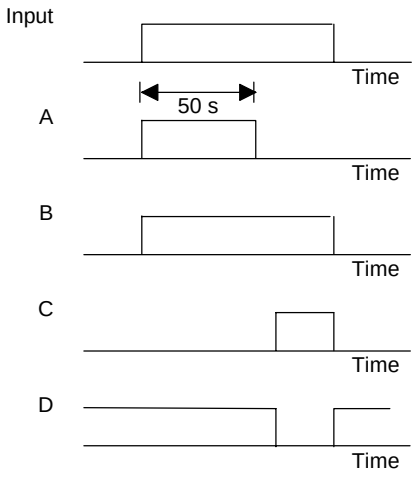


Figure 9.22 *Problems 16 to 19*

19 For the input shown in Figure 9.22, which is the output from O:013/04?

20 Devise ladder programs for systems that will carry out the following tasks:
(a) Switch on an output 5 s after receiving an input and keep it on for the duration of that input.
(b) Switch on an output for the duration of the input and then keep it on for a further 5 s.
(c) Switch on an output for 5 s after the start of an input signal.

10 Counters

Counters are provided as built-in elements in PLCs and allow the number of occurrences of input signals to be counted. This might be where items have to be counted as they pass along a conveyor belt, or the the number of revolutions of a shaft, or perhaps the number of people passing through a door. This chapter describes how such counters can be programmed.

10.1 Forms of counter

A counter is set to some preset number value and, when this value of input pulses has been received, it will operate its contacts. Thus normally open contacts would be closed, normally closed contacts opened.

There are two types of counter, though PLCs may not include both types. These are down-counters and up-counters. *Down-counters* count down from the preset value to zero, i.e. events are subtracted from the set value. When the counter reaches the zero value, its contacts change state. Most PLCs offer down counting. *Up-counters* count from zero up to the preset value, i.e. events are added until the number reaches the preset value. When the counter reaches the set value, its contacts change state.

Different PLC manufacturers deal with counters in slightly different ways. Some count down (CTD), or up (CTU), and reset and treat the counter as though it is a relay coil and so a rung output. In this way, counters can be considered to consist of two basic elements: one relay coil to count input pulses and one to reset the counter, the associated contacts of the counter being used in other rungs. Figure 10.1(a) illustrates this. Mitsubishi is an example of this type of manufacturer. Others treat the counter as an intermediate block in a rung from which signals emanate when the count is attained. Figure 10.1(b) illustrates this. Siemens is an example of this type of manufacturer.

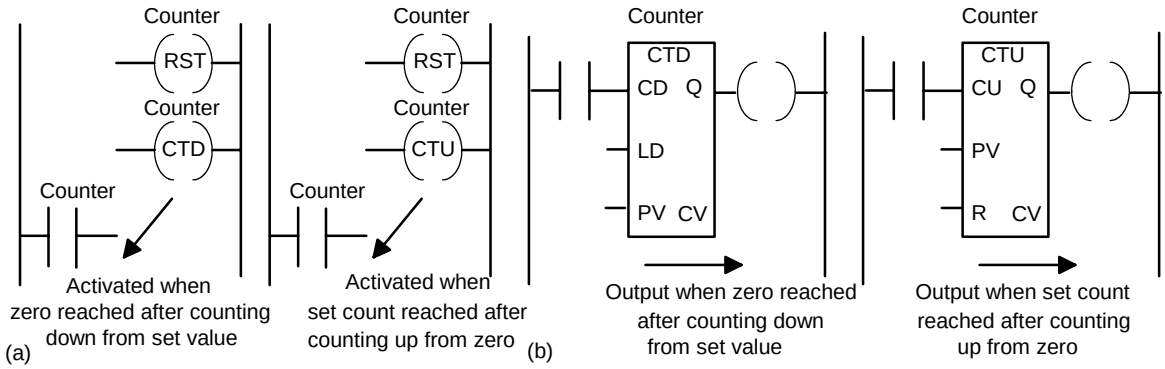


Figure 10.1 *Forms of representation of counters. In (a) RST is reset. In (b), the IEC 1131-3 representation, CD is count down input, LD is for loading the input, PV is for the preset value, CV the current count value, CU is count up input, and R is for the reset input.*

10.2 Programming

Figure 10.2 shows a basic counting circuit. When there is a pulse input to In 1, the counter is reset. When there is an input to In 2, the counter starts counting. If the counter is set for, say, 10 pulses, then when 10 pulse inputs have been received at In 2, the counter's contacts will close and there will be an output from Out 1. If at any time during the counting there is an input to In 1, the counter will be reset and start all over again and count for 10 pulses.

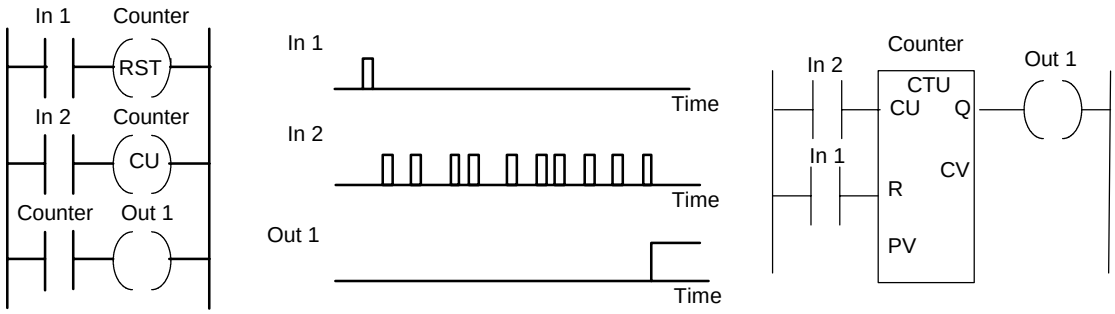


Figure 10.2 *Basic counter program*

Figure 10.3(a) shows how the above program, and its program instruction list, would appear with a Mitsubishi PLC. The reset and counting elements are combined in a single box spanning the two rungs. You can consider the rectangle to be enclosing the two counter ( ) outputs in Figure 10.2. The count value is set by a K program instruction. Figure 10.3(b) shows the same program with a Siemens PLC. With this ladder program, the counter is considered to be a delay element in the output line (as in Figure 10.1(b)). The counter is reset by an input to I0.1 and counts the pulses into input I0.0. The CU indicates that it is a count-up counter, a CD would indicate a count-down counter. The counter set value is indicated by the LKC number. Figure 10.3(c) is the program for Toshiba and Figure 10.3(d) for Allen-Bradley.

10.2.1 Counter application

As an illustration of the use that can be made of a counter, consider the problem of items passing along a conveyor belt, the passage of an item past a particular point being registered by a light beam to a photoelectric cell being interrupted, and after a set number there is to be a signal sent informing that the set count has been reached and the conveyor stopped. Figure 10.4(a) shows the basic elements of a Siemens program that could be used. A reset signal causes the counter to reset and start counting again. The set signal is used to make the counter active. Figure 10.4(b) shows the basic elements of the comparable Allen-Bradley program. When the count reaches the preset value, the done bit is set to 1 and so O:013/01 occurs and the corresponding contacts are opened and the conveyor stopped.

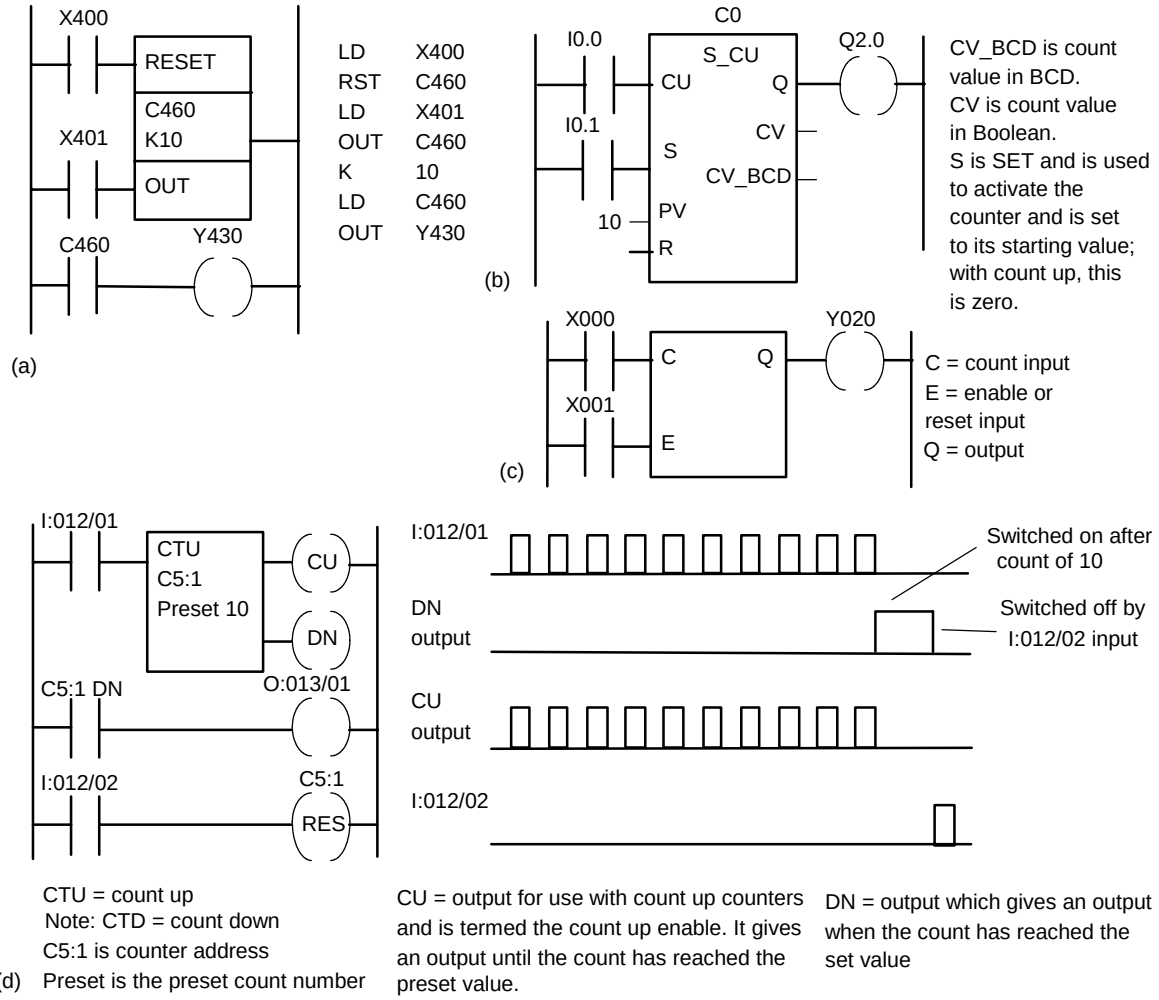


Figure 10.3 (a) *Mitsubishi program, (b) Siemens program, (c) Toshiba program, (b) Allen-Bradley programs*

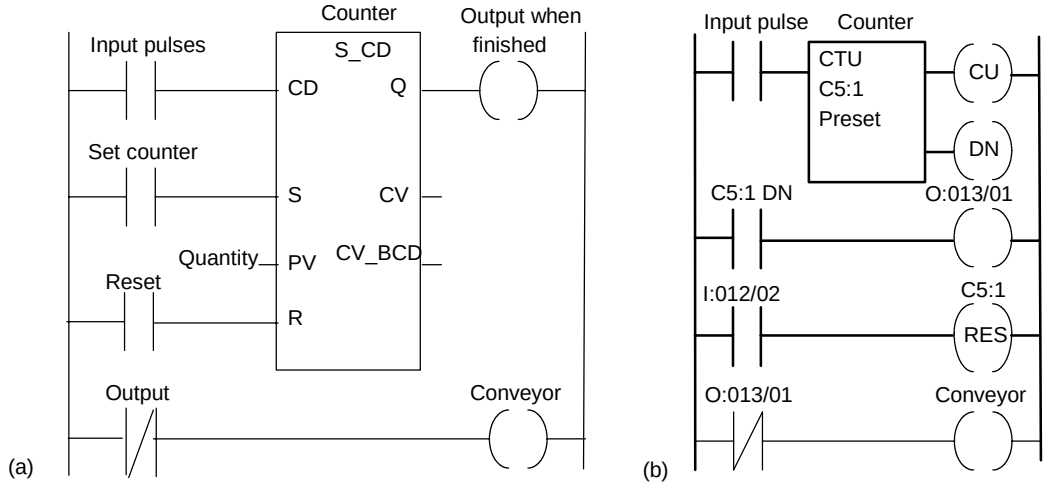


Figure 10.4 *(a) Siemens, (b) Allen-Bradley counting program*

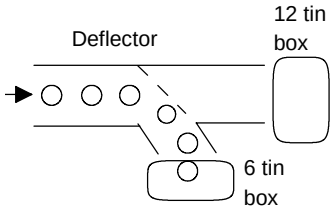
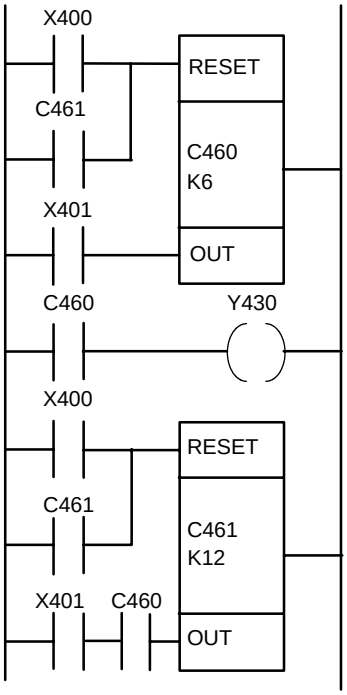


Figure 10.5 *Counting task*

As a further illustration of the use of a counter, consider the problem of the control of a machine which is required to direct 6 tins along one path for packaging in a box and then 12 tins along another path for packaging in another box (Figure 10.5). A deflector plate might be controlled by a photocell sensor which gives an output every time a tin passes it. Thus the number of pulses from the sensor has to be counted and used to control the deflector. Figure 10.6 shows the ladder program that could be used. Mitsubishi notation has been used.




```
LD    X400
OR    C461
RST   C460
K     6
LD    X401
OUT   C460
LD    C460
OUT   Y430
LD    X400
OR    C461
RST   C461
K     12
LD    X401
AND   C460
OUT   C461
```

Figure 10.6 *Ladder program for Figure 10.5 task*

When there is a pulse input to X400, both the counters are reset. The input to X400 could be the push button switch used to start the conveyor moving. The input which is counted is X401. This might be an input from a photocell sensor which detects the presence of tins passing along the conveyor. C460 starts counting after X400 is momentarily closed. When C460 has counted six items, it closes its contacts and so gives an output at Y430. This might be a solenoid which is used to activate a deflector to deflect items into one box or another. Thus the deflector might be in such a position that the first six tins passing along the conveyor are deflected into the 6-pack box, then the deflector plate is moved to allow tins to pass to the 12-pack box. When C460 stops counting it closes its contacts and so allows C461 to start counting. C461 counts for 12 pulses to X401 and then closes its contacts. This results in both counters being reset and the entire process can repeat itself.

Counters can be used to ensure that a particular part of a sequence is repeated a known number of times. This is illustrated by the following

program which is designed to enable a three-cylinder, double solenoid-controlled arrangement (Figure 10.7(a)) to give the sequence A+, A–, A+, A–, A+, A–, B+, C+, B–, C–. The A+, A– sequence is repeated three times before B+, C+, B–, C– occur. We can use a counter to enable this repetition. Figure 10.7(b) shows a possible program. The counter only allows B+ to occur after it has received three inputs corresponding to three a– signals.

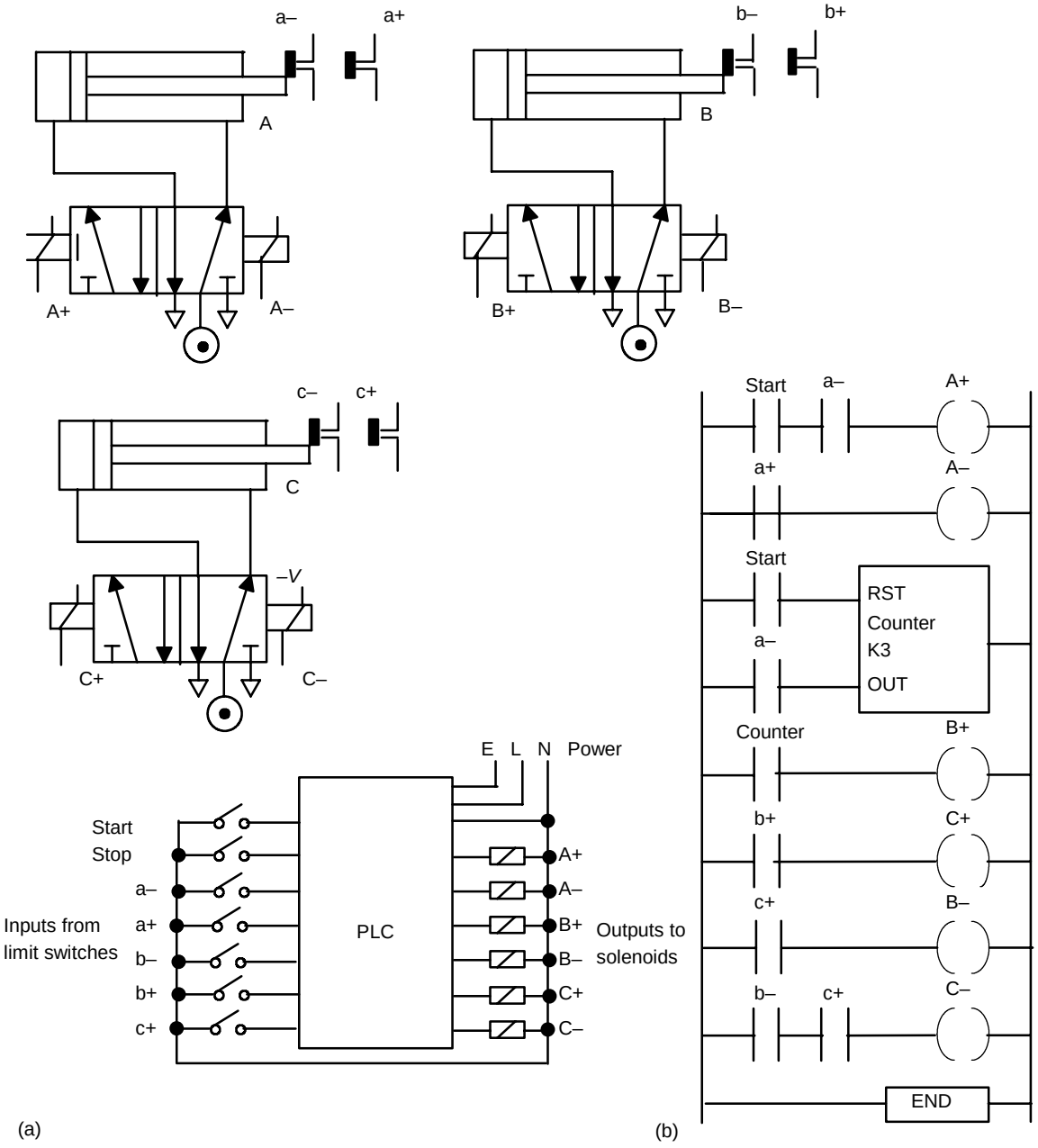


Figure 10.7 *(a) Three-cylinder system, (b) program*

10.3 Up and down counting

It is possible to program up- and down-counters together. Consider the task of counting products as they enter a conveyor line and as they leave it, or perhaps cars as they enter a multi-storage parking lot and as they leave it. An output is to be triggered if the number of items/cars entering is some number greater than the number leaving, i.e. the number in the parking lot has reached a 'saturation' value. The output might be to illuminate a 'No empty spaces' sign. Suppose we use the up-counter for items entering and the count down for items leaving. Figure 10.8(a) shows the basic form a ladder program for such an application can take. When an item enters it gives a pulse on input In 1. This increases the count by one. Thus each item entering increases the accumulated count by 1. When an item leaves it gives an input to In 2. This reduces the number by 1. Thus each item leaving reduces the accumulated count by 1. When the accumulated value reaches the preset value, the output Out 1 is switched on. Figure 10.8(b) shows the implementation of this program with an Allen-Bradley program.

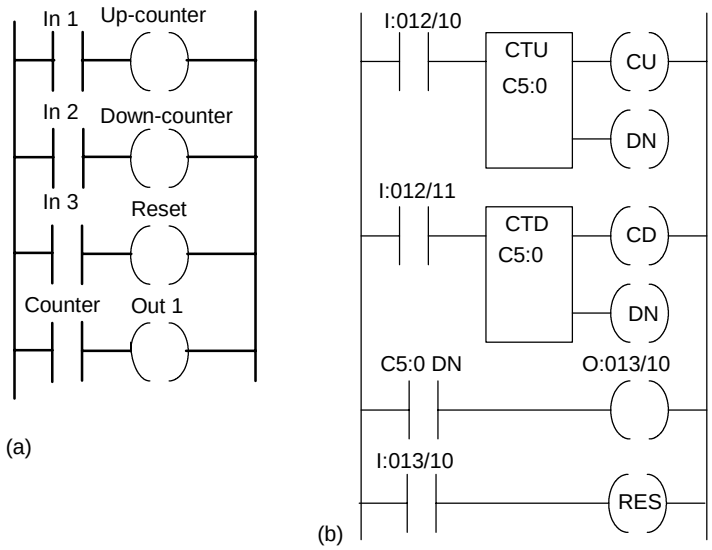


Figure 10.8 *(a) Using up- and down-counters, (b) Allen-Bradley program*

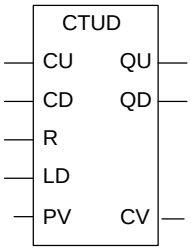


Figure 10.9 *IEC 1131-3 standard symbol for up-down counter*

Up-down counters are available as single entities. Figure 10.9 shows the IEC 1131-3 standard symbol. The counter has two inputs CU and CD and counts up the number of pulses detected at the input CU and counts down the number of pulses detected at input CD. If the counter input reaches zero, the QD output is set on and the counting down stops. If the count reaches the maximum value PV, the QU output is set on and the counting up stops. CV is the count value. LD can be used to preset the counter output CV with the value PV. The reset R clears the counter input to zero.

Figure 10.10 shows how the above system might appear for a Siemens PLC and the associated program instruction list. CU is the count up input and CD the count down. R is the reset. The set accumulator value is loaded via F0.0, this being an internal relay.

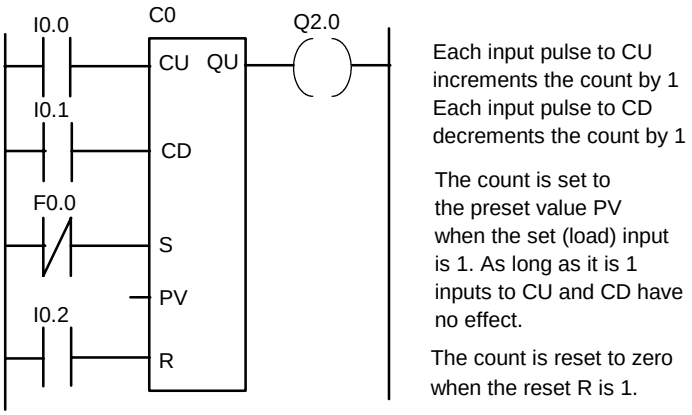

I0.0
C0
Q2.0
CU QU
I0.1
CD
F0.0
S
PV
I0.2
R
Each input pulse to CU increments the count by 1
Each input pulse to CD decrements the count by 1
The count is set to the preset value PV when the set (load) input is 1. As long as it is 1 inputs to CU and CD have no effect.
The count is reset to zero when the reset R is 1.

Figure 10.10 *Up and down counting with a Siemens PLC*

10.4 Timers with counters

A typical timer can count up to 16 binary bits of data, this corresponding to 32 767 base time units. Thus, if we have a time base of 1 s then the maximum time that can be dealt with by a timer is just over 546 minutes or 9.1 hours. If the time base is to be 0.1 s then the maximum time is 54.6 minutes or just short of an hour. By combining a timer with a counter, longer times can be counted. Figure 10.11 illustrates this with an Allen-Bradley program. If the timer has a time base of 1 s and a preset value of 3600, then it can count for up to 1 hour. When input I:012/01 is activated, the timer starts to time in one second increments. When the time reaches the preset value of 1 hour, the DN bit is set to 1 and the counter increments by 1. The DN bit setting to 1 also reset the timer and the timer starts to time again. When it next reaches its preset time of 1 hour, the DN bit is again set to 1 and the counter increments by 1. With the counter set to a preset value of 24, the counter DN bit is set to 1 when the count reaches 24 and the output O:013/01 is turned on. We thus have a timer which is able to count the seconds for the duration of a day and would be able to switch on some device after 24 hours.

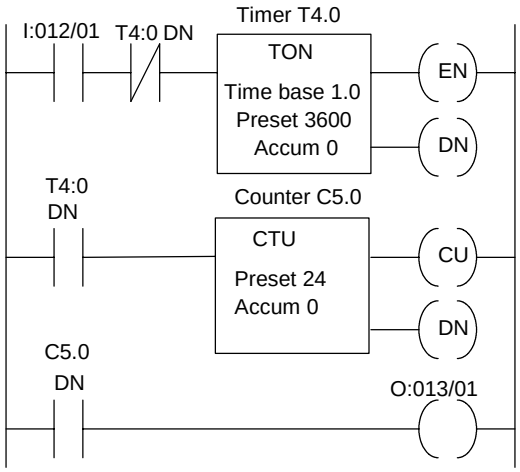


Figure 10.11 *Using a counter to extend the range of a timer*

10.5 Sequencer

The *drum sequencer* is a form of counter that is used for sequential control. It replaces the mechanical drum sequencer that was used to control machines that have a stepped sequence of repeatable operations. One form of the mechanical drum sequencer consisted of a drum from which a number of pegs protruded (Figure 10.12). When the cylinder rotated, contacts aligned with the pegs were closed when the peg impacted on them and opened when the peg had passed. Thus for the arrangement shown in Figure 10.12, as the drum rotates, in the first step the peg for output 1 is activated, in step 2 the peg for the third output, in step 3 the peg for the second output, and so on. Different outputs could be controlled by pegs located at different distances along the drum. Another form consisted of a series of cams on the same shaft, the profile of the cam being used to switch contacts on and off.

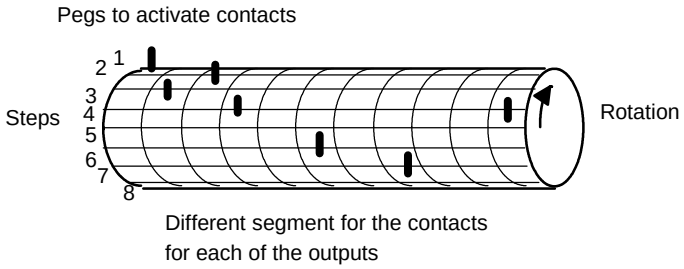


Figure 10.12 *Drum sequencer*

The PLC sequencer consists of a master counter that has a range of preset counts corresponding to the different steps and so, as it progresses through the count, when each preset count is reached it can be used to control outputs. Each step in the count sequence relates to a certain output or group of outputs. The outputs are internal relays, these in turn being used to control the external output devices.

Suppose we want output 1 to be switched on 5 s after the start and remain on until the time reaches 10 s, output 2 to be switched on at 10 s and remain on until 20 s, output 3 to be switched on at 15 s and remain on until 25 s, etc. We can represent the above requirements by a time sequence diagram, Figure 10.13, showing the required time sequence.

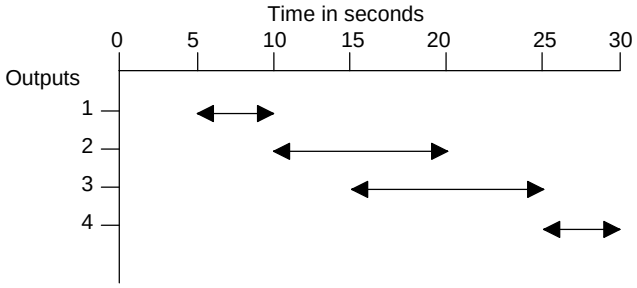


Figure 10.13 *Timing diagram*

We can transform the timing diagram into a drum sequence requirement. Taking each drum sequence step to take 5 s gives the

requirement diagram shown in Table 10.1. Thus at step 1 we require output 1 to be switched on and to remain on until step 2. At step 2 we require output 2 to be switched on and remain on until step 4. At step 3 we require output 3 to be switched on and remain on until step 5. At step 5 we require output 4 to be switched on and remain on until step 6.

Table 10.1 *Sequence requirements*

Step	Time (s)	Output 1	Output 2	Output 3	Output 4
0	0	0	0	0	0
1	5	1	0	0	0
2	10	0	1	0	0
3	15	0	1	1	0
4	20	0	0	1	0
5	25	0	0	1	1
6	30	0	0	0	0

With a PLC, such as a Toshiba, the sequencer is set up by switching on the Step Sequence Initialize (STIZ) function block R500 (Figure 10.14). This sets up the program for step 1 and R501. This relay then switches on output Y020. The next step is the switching on of R502. This switches on the output Y021 and also a delay-on timer so that R503 is not switched on until the timer has timed out. Then R503 switches on Y022 and also the next step in the sequence.

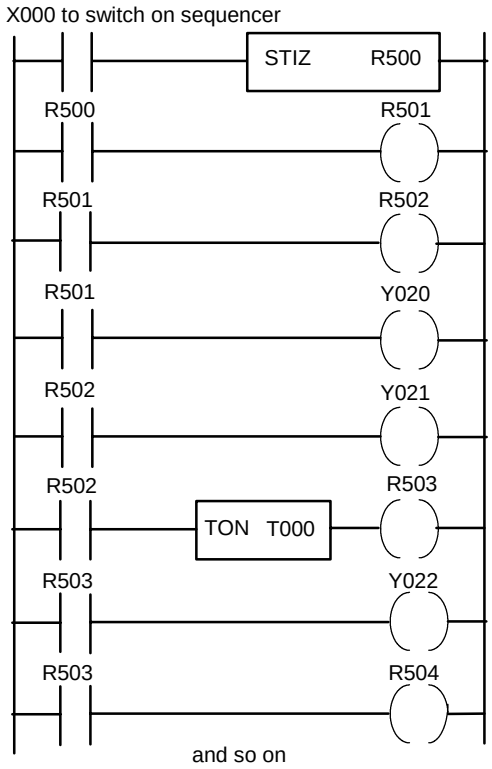


Figure 10.14 *Sequencer with a Toshiba PLC*

With the Allen-Bradley form of PLC the sequencer is programmed by using a sequence of binary words in the form of the outputs required, e.g. those listed in Table 10.1. Thus we would have the following binary word sequence put into the program using the programming device.

```
Input 4
|  Input 3
|  |  Input 2
|  |  |  Input 1
|  |  |  |
0000
0001
0010
0110
0100
0100
0000
```

Problems

Questions 1 to 19 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

Problems 1 to 3 refer to Figure 10.15 which shows a ladder diagram with a counter, two inputs In 1 and In 2 and an output Out 1.

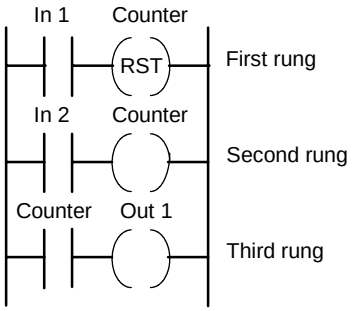


Figure 10.15 *Problems 1 to 3*

1 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 10.15, when the counter is set to 5, there is an output from Out 1 every time:
(i) In 1 has closed 5 times.
(ii) In 2 has closed 5 times.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

2 Decide whether each of these statements is True (T) or False (F).

For the ladder diagram shown in Figure 10.15:
(i) The first rung gives the condition required to reset the counter.
(ii) The second rung gives the condition required to generate pulses to be counted.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

3 Decide whether each of these statements is True (T) or False (F).

In Figure 10.15, when there is an input to In 1:
(i) The counter contacts in the third rung close.
(ii) The counter is ready to start counting the pulses from In 2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 4 and 5 refer to the following program instruction list involving a down-counter:

```
LD      X400
RST     C460
LD      X401
OUT     C460
K       5
LD      460
OUT     Y430
```

4 Decide whether each of these statements is True (T) or False (F).

Every time there is an input to X401:
(i) The count accumulated by the counter decreases by 1.
(ii) The output is switched on.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

5 Decide whether each of these statements is True (T) or False (F).

When there is an input to X400, the counter:
(i) Resets to a value of 5.
(ii) Starts counting from 0.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 6 and 7 refer to the following program instruction list involving a counter C0:

```
A       I0.0
CD      C0
LKC     5
A       I0.1
R       C0
Q       2.00
```

6 Decide whether each of these statements is True (T) or False (F).

Every time there is an input to I0.0:
(i) The count accumulated by the counter decreases by 1.
(ii) The output is switched on.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

7 Decide whether each of these statements is True (T) or False (F).

When there is an input to I0.1, the counter:
(i) Resets to a value of 5.
(ii) Starts counting from 0.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 8 and 9 refer to Figure 10.16 which shows a down-counter C460 controlled by two inputs X400 and X401, there being an output from Y430.

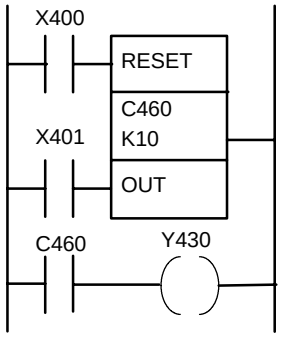


Figure 10.16 *Problems 8 and 9*

8 Decide whether each of these statements is True (T) or False (F).

When there is an input to X400, the counter:
(i) Resets to a value of 0.
(ii) Starts counting.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

9 Decide whether each of these statements is True (T) or False (F).

Every time there is an input to X401, the counter:
(i) Gives an output from Y430.
(ii) Reduces the accumulated count by 1.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 10 to 12 refer to Figure 10.17 which shows a ladder diagram involving a counter C460, inputs X400 and X401, internal relays M100 and M101, and an output Y430.

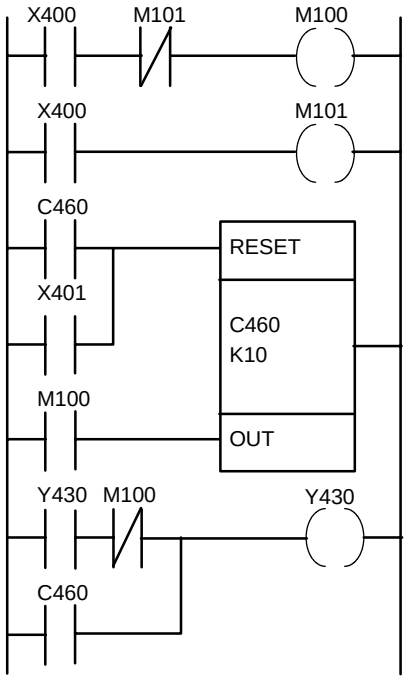


Figure 10.17 *Problems 10 to 12*

10 Decide whether each of these statements is True (T) or False (F).

For the output Y430:
(i) It switches on the tenth pulse to X400.
(ii) It switches off at the start of the eleventh pulse to X400.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

11 Decide whether each of these statements is True (T) or False (F).

When there is an input to X400:
(i) The internal relay M100 is energised.
(ii) The internal relay M101 is energised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

12 Decide whether each of these statements is True (T) or False (F).

There is an output from Y430 as long as:
(i) The C460 contacts are closed.
(ii) Y430 gives an output and M100 is energised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

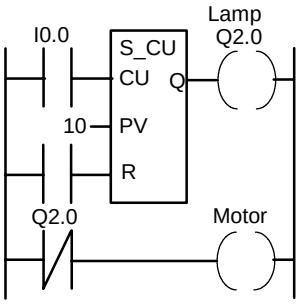


Figure 10.18 *Problem 13*

13 Decide whether each of these statements is True (T) or False (F).

Figure 10.18 shows a counter program in Siemens format. After 10 inputs to I0.0:
(i) The lamp comes on.
(ii) The motor starts.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 14 and 15 refer to Figure 10.19 which shows a Siemens program involving an up- and down-counter.

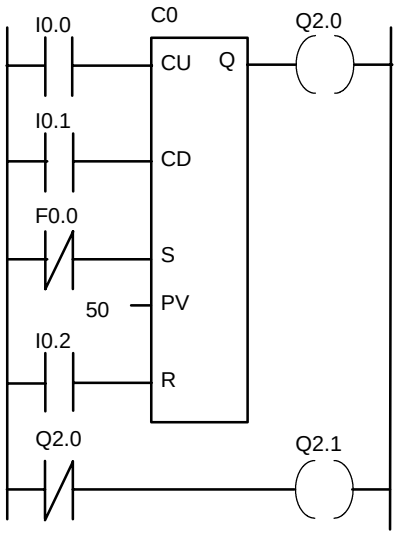


Figure 10.19 *Problems 14 and 15*

14 Decide whether each of these statements is True (T) or False (F).

When the count is less than 50:
(i) There is an output from Q2.0.
(ii) There is an output from Q2.1.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

15 Decide whether each of these statements is True (T) or False (F).

When the count reaches 50:
(i) There is an output from Q2.0.
(ii) There is an output from Q2.1.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 16 and 17 refer to Figure 10.20 which shows a Allen-Bradley program involving an count-up counter.

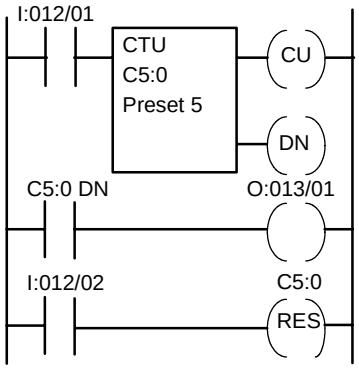


Figure 10.20 *Problems 16 and 17*

16 For the program shown in Figure 10.20, the counter is reset when:

A The count reaches 5.
B The count passes 5.
C There is an input to I:012/01.
D There is an input to I:012/02.

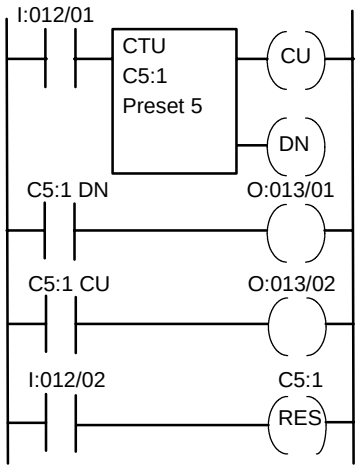


Figure 10.21 *Problems 18 and 19*

17 Decide whether each of these statements is True (T) or False (F).

For the program shown in Figure 10.20, there is an output at O:013/01 when:
(i) There is an input to I:012/01.
(ii) There is an output from the count up done bit DN.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 18 and 19 refer to Figure 10.21 which shows a Allen-Bradley program involving a count-up counter.

18 Decide whether each of these statements is True (T) or False (F).

When there is a single pulse input to I:012/01:

(i) Output O:013/01 is switched on.
(ii) Output O:013/02 is switched on.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

19 Decide whether each of these statements is True (T) or False (F).

When the fifth pulse input occurs to I:012/01:
(i) Output O:013/01 is switched on.
(ii) Output O:013/02 is switched on.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

20 Devise ladder programs for systems that will carry out the following tasks:
(a) Give an output after a photocell sensor has given 10 pulse input signals as a result of detecting 10 objects passing in front of it.
(b) Give an output when the number of people in a store reaches 100, there continually being people entering and leaving the store.

11 Shift registers

The term *register* is used for an electronic device in which data can be stored. An internal relay, see Chapter 7, is such a device. The *shift register* is a number of internal relays grouped together which allow stored bits to be shifted from one relay to another. This chapter is about shift registers and how they can be used where a sequence of operations is required or to keep track of particular items in a production system.

11.1 Shift registers

A register is a number of internal relays grouped together, normally 8, 16 or 32. Each internal relay is either effectively open or closed, these states being designated as 0 and 1. The term *bit* is used for each such binary digit. Therefore, if we have eight internal relays in the register we can store eight 0/1 states. Thus we might have:

Internal relays

1	2	3	4	5	6	7	8

and each relay might store an on-off signal such that the state of the register at some instant is:

1	0	1	1	0	0	1	0

i.e. relay 1 is on, relay 2 is off, relay 3 is on, relay 4 is on, relay 5 is off, etc. Such an arrangement is termed an 8-bit register. Registers can be used for storing data that originate from input sources other than just simple, single on-off devices such as switches.

With the *shift register* it is possible to shift stored bits. Shift registers require three inputs, one to load data into the first location of the register, one as the command to shift data along by one location and one to reset or clear the register of data. To illustrate this, consider the following situation where we start with an 8-bit register in the following state:

1	0	1	1	0	0	1	0

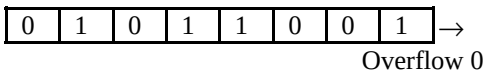
Suppose we now receive the input signal 0. This is an input signal to the first internal relay.

Input 0

$\rightarrow$

1	0	1	1	0	0	1	0

If we also receive the shift signal, then the input signal enters the first location in the register and all the bits shift along one location. The last bit overflows and is lost.



Thus a set of internal relays that were initially on, off, on, on, off, off, on, off are now off, on, off, on, on, off, off, on.

The grouping together of internal relays to form a shift register is done automatically by a PLC when the shift register function is selected. With the Mitsubishi PLC, this is done by using the programming code SFT (shift) against the internal relay number that is to be the first in the register array. This then causes a block of relays, starting from that initial number, to be reserved for the shift register.

11.2 Ladder programs

Consider a 4-bit shift register and how it can be represented in a ladder program (Figure 11.1(a)). The input In 3 is used to reset the shift register, i.e. put all the values at 0. The input In 1 is used to input to the first internal relay in the register. The input In 2 is used to shift the states of the internal relays along by one. Each of the internal relays in the register, i.e. IR 1, IR 2, IR 3 and IR 4, is connected to an output, these being Out 1, Out 2, Out 3 and Out 4.

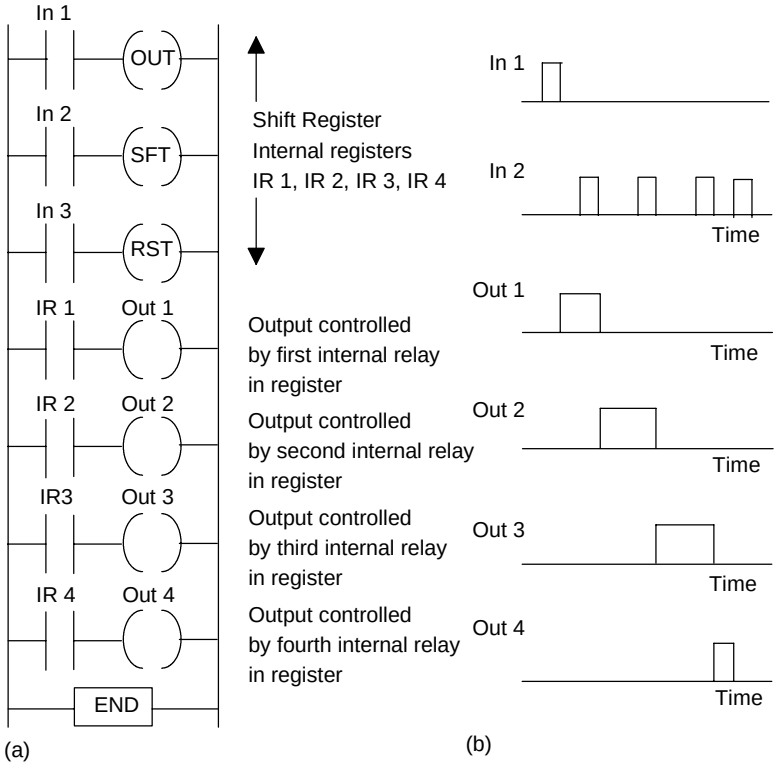


Figure 11.1 *The shift register*

Suppose we start by supplying a momentary input to In 3. All the internal relays are then set to 0 and so the states of the four internal relays IR 1, IR 2, IR 3 and IR 4 are 0, 0, 0, 0. When In 1 is momentarily closed there is a 1 input into the first relay. Thus the states of the internal relays IR 1, IR 2, IR 3 and IR 4 are now 1, 0, 0, 0. The IR 1 contacts close and we thus end up with an output from Out 1. If we now supply a momentary input to In 2, the 1 is shifted from the first relay to the second. The states of the internal relays are now 0, 1, 0, 0. We now have no input from Out 1 but an output from Out 2. If we supply another momentary input to In 2, we shift the states of the relays along by one location to give 0, 0, 1, 0. Outputs 1 and 2 are now off but Out 3 is on. If we supply another momentary input to In 2 we again shift the states of the relays along by one and have 0, 0, 0, 1. Thus now, outputs 1, 2 and 3 are off and output 4 has been switched on. When another momentary input is applied to In 2, we shift the states of the relays along by one and have 0, 0, 0, 0 with the 1 overflowing and being lost. All the outputs are then off. Thus the effect of the sequence of inputs to In 2 has been to give a sequence of outputs Out 1, followed by Out 2, followed by Out 3, followed by Out 4. Figure 11.1(b) shows the sequence of signals.

Figure 11.2 shows the Mitsubishi version of the above ladder program and the associated instruction list. Instead of the three separate outputs for reset, output and shift, the Mitsubishi shift register might appear in a program as a single function box, as shown in the Figure. With the Mitsubishi shift register, the M140 is the address of the first relay in the register.

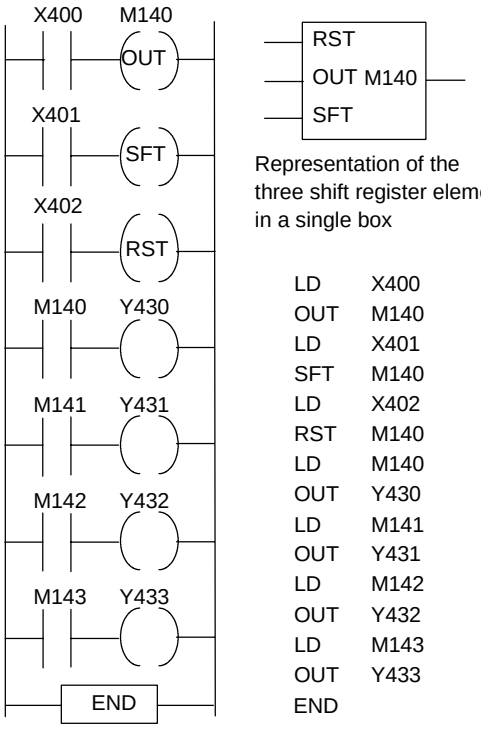


Figure 11.2 *Mitsubishi program*

Figure 11.3 shows a shift register ladder program for a Toshiba PLC. With the Toshiba R016 is the address of the first relay in the register. The (08) indicates that there are eight such relays. D is used for the data input, S for shift input, E for enable or reset input and Q for output.

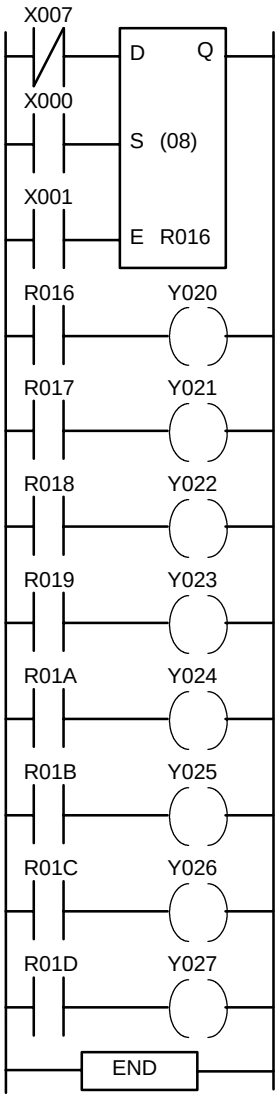


Figure 11.3 *Shift register*

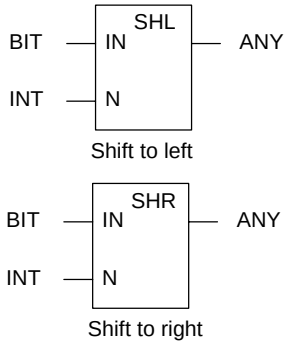


Figure 11.4 *IEC 1131-3 shift register symbols*

Figure 11.4 shows the IEC 1131-3 standard symbol for a shift register. The value to be shifted is at input IN and the number of places it is to be shifted is at input N.

Figure 11.5 shows the Siemens symbol for a shift register. If the enable input EN 1 the shift function is executed and ENO is then 1. If EN is 0 the shift function is not executed and ENO is 0. The shift function SHL_W shifts the contents of the word variable at input IN bit by bit to the left the number of positions specified by the input at N. The shifted word output is at OUT.

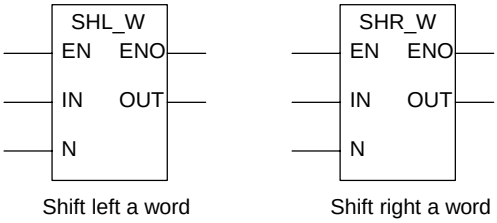


Figure 11.5 *Siemens shift register symbol*

11.2.1 A sequencing application

Consider the requirement for a program for two double solenoid cylinders, the arrangement being as shown in Figure 11.6(a), to give the sequence A+, B+, A–, B–. Figure 11.6(b) shows a program to achieve this sequence by the use of a shift register.

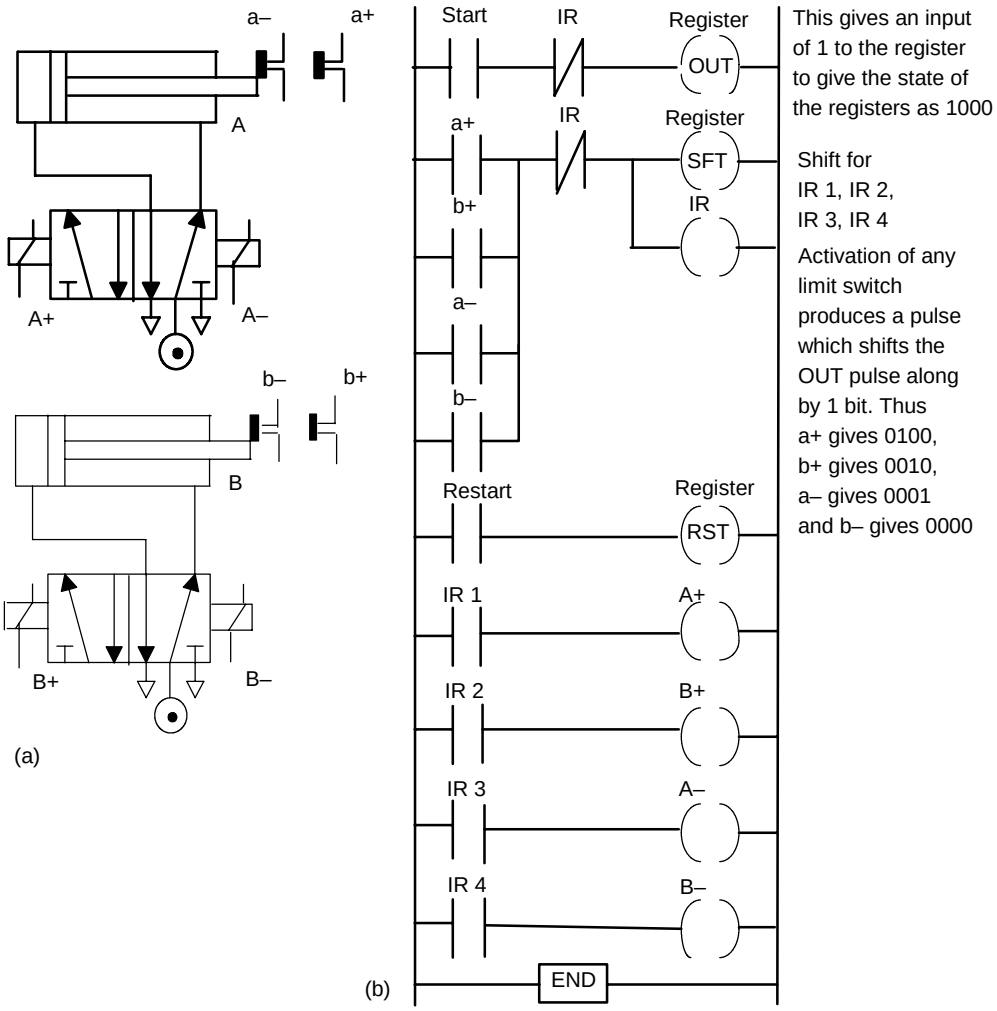


Figure 11.6 *Sequencing cylinders*

11.2.2 Keeping track of items

The above indicates how a shift register can be used for sequencing. Another application is to keep track of items. For example, a sensor might be used to detect faulty items moving along a conveyor and keep track of it so that when it reaches the appropriate point a reject mechanism is activated to remove it from the conveyor. Figure 11.7 illustrates this arrangement and the type of ladder program that might be used.

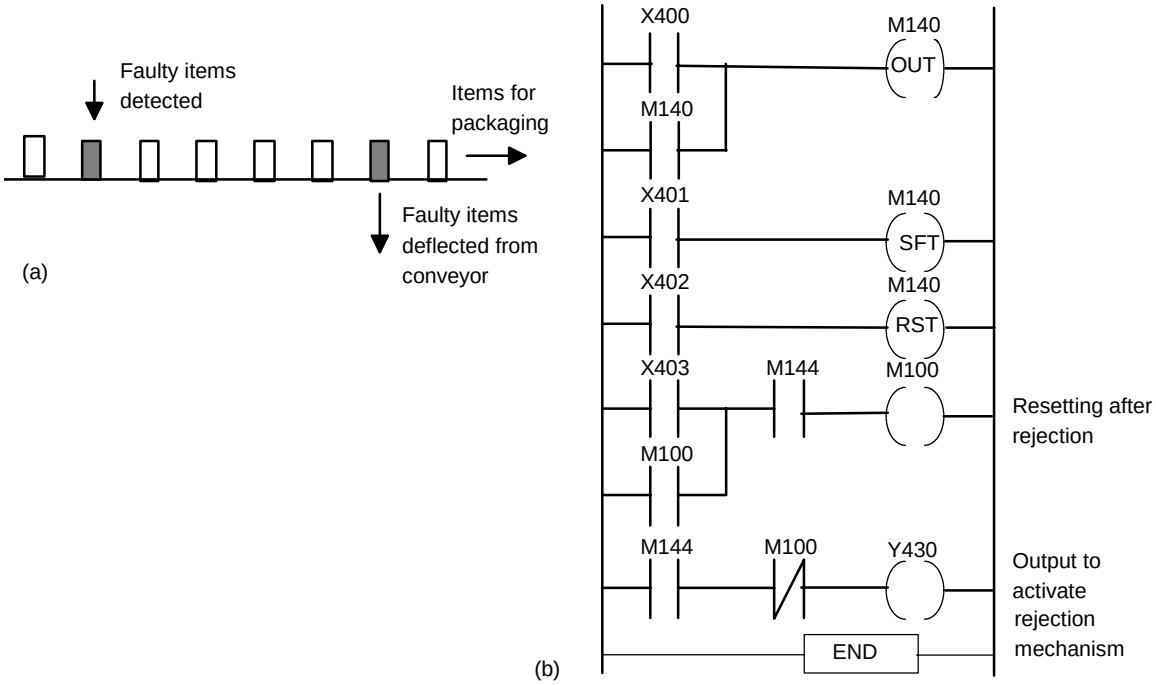


Figure 11.7 *Keeping track of faulty items*

Each time a faulty item is detected, a pulse signal occurs at input X400. This enters a 1 into the shift register at internal relay M140. When items move, whether faulty or not, there is a pulse input at X401. This shifts the 1 along the register. When the 1 reaches internal relay M144, it activates the output Y430 and the rejection mechanism removes the faulty item from the conveyor. When an item is removed it is sensed and an input to X403 occurs. This is used to reset the mechanism so that no further items are rejected until the rejection signal reaches M144. It does this by giving an output to internal relay M100 which latches the X403 input and switches the rejection output Y430 off. This represents just the basic elements of a system. A practical system would include further internal relays in order to make certain that the rejection mechanism is off when good items move along the conveyor belt and also to disable the input from X400 when the shifting is occurring.

Problems

Questions 1 to 9 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

Problems 1 to 5 concern a 4-bit shift register, involving internal relays IR 1, IR 2, IR 3 and IR 4, which has been reset to 0, 0, 0, 0.

1 When there is a pulse 1 input to the OUT of the shift register, the internal relays in the shift register show:

A 0001
B 0010
C 0100
D 1000

2 Following a pulse input of 1 to the OUT of the shift register, there is a pulse input to SHIFT. The internal relays then show:

A 0001
B 0010
C 0100
D 1000

3 With a continuous input of 1 to the OUT of the shift register, there is a pulse input to SHIFT. The internal relays then show:

A 0011
B 0110
C 1100
D 0010

4 With a continuous input of 1 to the OUT of the shift register, there are two pulse inputs to SHIFT. The internal relays then show:

A 0001
B 0010
C 1100
D 1110

5 With a pulse input of 1 to the OUT of the shift register, there is a pulse input to SHIFT, followed by a pulse input to RESET. The internal relays then show:

A 0000
B 0010
C 0100
D 1000

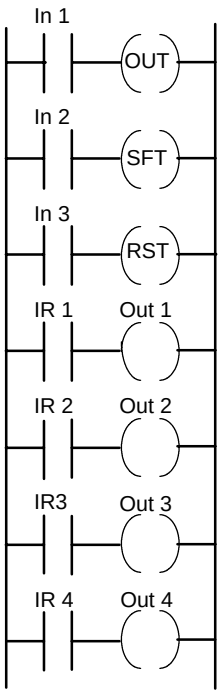


Figure 11.8 *Problems 6 to 9*

Problems 6 to 9 concern Figure 11.8 which shows a 4-bit shift register with internal relays IR 1, IR 2, IR 3 and IR 4, with three inputs In 1, In 2 and In 3, and four outputs Out 1, Out 2, Out 3 and Out 4.

6 Decide whether each of these statements is True (T) or False (F).

When there is a pulse input to In 1:
(i) The output Out 1 is energised.
(ii) The contacts of the internal relay IR 1 close.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

7 Decide whether each of these statements is True (T) or False (F).

When there is a pulse input to In 1 followed by a pulse input to SFT:
(i) The output Out 1 is energised.
(ii) The output Out 2 is energised.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

8 Decide whether each of these statements is True (T) or False (F).

To obtain the outputs Out 1, Out 2, Out 3 and Out 4 switching on in sequence and remaining on, we can have for inputs:
(i) A pulse input to In 1 followed by three pulse inputs to SFT.
(ii) A continuous input to In 1 followed by three pulse inputs to SFT.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

9 Initially: Out 1 off, Out 2 off, Out 3 off, Out 4 off
Next: Out 1 on, Out 2 off, Out 3 off, Out 4 off
Next: Out 1 off, Out 2 on, Out 3 off, Out 4 off
Next: Out 1 on, Out 2 off, Out 3 on, Out 4 off

The inputs required to obtain the above sequence are:

A Pulse input to In 1 followed by pulse input to In 2.
B Pulse input to In 1 followed by two pulses to In 2.
C Pulse input to In 1 followed by pulse input to In 2, followed by pulse input to In 1.
D Pulse input to In 1 followed by pulse input to In 2, followed by pulse inputs to In 1 and In 2.

10 Devise ladder programs for systems to carry out the following tasks:
(a) A sequence of four outputs such that output 1 is switched on when the first event is detected and remains on, output 2 is switched on when the second event is detected and remains on, output 3 is switched on when the third event is detected and remains on, output 4 is switched on when the fourth event is detected and remains on, and all outputs are switched off when one particular input signal occurs.
(b) Control of a paint sprayer in a booth through which items pass on an overhead conveyor so that the paint is switched on when a part is in front of the paint gun and off when there is no part. The items are suspended from the overhead conveyor by hooks, not every hook having an item suspended from it.

12 Data handling

Timers, counters and individual internal relays are all concerned with the handling of individual bits, i.e. single on-off signals. Shift registers involve a number of bits with a group of internal relays being linked (see Chapter 11). The block of data in the register is manipulated. This chapter is about PLC operations involving blocks of data representing a value, such blocks being termed *words*. A block of data is needed if we are to represent numbers rather than just a single on-off input. Data handling consists of operations involving moving or transferring numeric information stored in one memory word location to another word in a different location, comparing data values and carrying out simple arithmetic operations. For example, there might be the need to compare a numeric value with a set value and initiate action if the actual value is less than the set value. This chapter is an introductory discussion of such operations.

12.1 Registers and bits

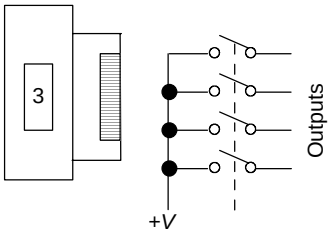
A register is where data can be stored (see Section 8.1 for an initial discussion of registers). In a PLC there are a number of such registers. Each data register can store a *binary word* of usually 8 or 16 bits. The number of bits determines the size of the number that can be stored. The *binary system* uses only two symbols, 0 and 1 (see Chapter 3). Thus we might have the 4-bit number 1111. This is the denary number, i.e. the familiar number system based on 10s, of $2^0 + 2^1 + 2^2 + 2^3 = 1 + 2 + 4 + 8 = 15$. Thus a 4-bit register can store a positive number between 0 and $2^0 + 2^1 + 2^2 + 2^3$ or $2^4 - 1 = 15$. An 8-bit register can store a positive number between 0 and $2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7$ or $2^8 - 1$, i.e. 255. A 16-bit register can store a positive number between 0 and $2^{16} - 1$, i.e. 65 535.

Thus a 16-bit word can be used for positive numbers in the range 0 to +65 535. If negative numbers are required, the most significant bit is used to represent the sign, a 1 representing a negative number and a 0 a positive number and the format used for the negative numbers is *two's complement*. Two's complement is a way of writing negative numbers so that when we add, say, the signed equivalent of +5 and –5 we obtain 0. Thus in this format, 1011 represents the negative number –5 and 0101 the positive number +5; 1011 + 0101 = (1)0000 with the (1) for the 4-bit number being lost. See Chapter 3 for further discussion.

The *binary coded decimal* (BCD) format is often used with PLCs when they are connected to devices such as digital displays. With the natural binary number there is no simple link between the separate symbols of a denary number and the equivalent binary number. You have to work out the arithmetic to establish one number from the other. With the binary

coded decimal system, each denary digit is represented, in turn, by a 4-bit binary number (four is the smallest number of binary bits that gives a denary number greater than 10, i.e. $2^n > 10$). To illustrate this, consider the denary number 123. The 3 is represented by the 4-bit binary number 0011, the 2 by the 4-bit number 0010 and the 1 by 0001. Thus the binary coded decimal number of 123 is 0001 0010 0011. Binary coded decimal is a convenient system for use with external devices which are arranged in denary format, e.g. decade switches (thumbwheel switches) and digital displays. Then four binary bits can be used for each denary digit. PLCs therefore often have inputs or outputs which can be programmed to convert binary coded decimal from external input devices to the binary format needed for inside the PLC and from the binary format used internally in the PLC to binary coded decimal for external output devices (see Section 12.3).

The thumbwheel switch is widely used as a means of inputting BCD data manually into a PLC. It has four contacts which can be opened or closed to give the four binary bits to represent a denary number (Figure 12.1). The contacts are opened or closed by rotating a wheel using one's thumb. By using a number of such switches, data can be inputted in BCD format.



Position	Switch outputs
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

0 = switch open 1 = switch closed

Figure 12.1 *Thumbwheel switch*

12.2 Data handling

The following are examples of data-handling instructions to be found with PLCs.

12.2.1 Data movement

The structured text program entry for a MOVE operation is :- with the destination address given first and then after :- the value to be moved to the destination, i.e.

Result :- value

With ladder programs, Figure 12.2 illustrates a common practice of using one rung of a ladder program for each move operation, showing the form used by three manufacturers, Mitsubishi, Allen-Bradley, and Siemens. For the rung shown, when there is an input to | | in the rung, the

move occurs from the designated source address to the designated destination address. For data handling with these PLCs, the typical ladder program data-handling instruction contains the data-handling instruction, the source (S) address from where the data is to be obtained and the destination (D) address to where it is to be moved. The approach that is used by some manufacturers, e.g. Siemens, is to regard data movement as two separate instructions, loading data from the source into an accumulator and then transferring the data from the accumulator to the destination. Figure 12.2(c) shows the Siemens symbol for the MOVE function. The data is moved from the IN input to the OUT output when EN is enabled.

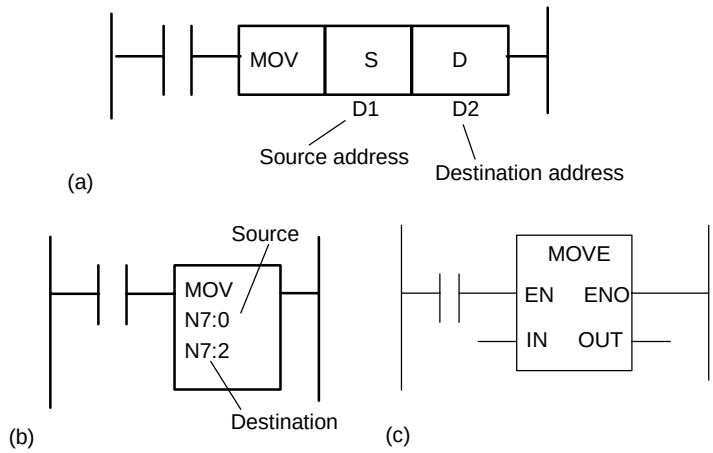


Figure 12.2 *Data movement: (a) Mitsubishi, (b) Allen-Bradley, (c) Siemens*

Data transfers might be to move a preset value to a timer or counter, or a time or counter value to some register for storage, or data from an input to a register or a register to output. Figure 12.3 shows the rung, in the Allen-Bradley format, that might be used to transfer a number held at address N7:0 to the preset of timer T4:6 when the input conditions for that rung are met. A data transfer from the accumulated value in a counter to a register would have a source address of the form C5:18.ACC and a destination address of the form N7:0. A data transfer from an input to a register might have a source address of the form I:012 and a destination address of the form N7:0. A data transfer from a register to an output might have a source address of the form N7:0 and a destination address of the form O:030.

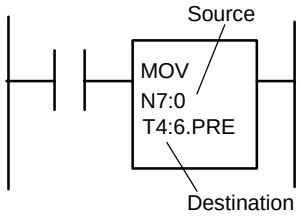


Figure 12.3 *Moving number to timer preset*

12.2.2 Data comparison

The data comparison instruction gets the PLC to compare two data values. Thus it might be to compare a digital value read from some input device with a second value contained in a register. For example, we might want some action to be initiated when the input from a temperature sensor gives a digital value which is less than a set value stored in a data register in the PLC. PLCs generally can make comparisons for *less than* (< or LT or LES), *equal to* (= or = = or EQ or EQU), *less than or equal to* (≤ or <= or LE or LEQ), *greater than* (> or GT or GRT), *greater than or equal to* (≥ or >= or GE or GEQ) and *not equal to* (≠ or <> or NE or NEQ). The brackets alongside each of the terms indicates common abbreviations used in programming. As an illustration, in structured text we might have:

```
(*Check that boiler pressure P2 is less than pressure P1*)
Output :- P2 < P1;
```

With ladder programs, for data comparison the typical instruction will contain the data-transfer instruction to compare data, the source (S) address from where the data is to be obtained for the comparison and the destination (D) address of the data against which it is to be compared. The instructions commonly used for the comparison are the terms indicated in the above brackets. Figure 12.4 shows the type of formats used by two manufacturers using the greater than form of comparison. Similar forms apply to the other forms of comparison. In Figure 12.4(a) the format is that used by Mitsubishi, S indicating the source of the data value for the comparison and D the destination or value against which the comparison is to be made. Thus if the source value is greater than the destination value, the output is 1. In Figure 12.4(b) the Allen-Bradley format has been used. Here the source of the data being compared is given as the accumulated value in timer 4.0 and the data against which it is being compared is the number 400. Figure 12.4(c) shows the Siemens format. The values to be compared are at inputs IN1 and IN2 and the result of the comparison is at the output, being 1 if the comparison is successfull otherwise 0. The R is used to indicate real numbers, i.e. floating point numbers, I being used for integers, i.e. fixed-point numbers involving 16 bits, and D for fixed-point numbers involving 32 bits. Both the inputs need to be of the same data type, e.g. REAL.

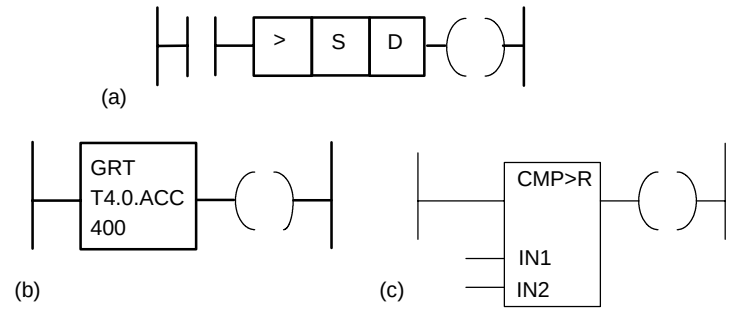


Figure 12.4 *Greater than comparison: (a) Mitsubishi, (b) Allen-Bradley, (c) Siemens*

As an illustration of the use of such a comparison, consider the task of sounding an alarm if a sensor indicates that a temperature has risen above some value, say 100ºC. The alarm is to remain sounding until the temperature falls below 90ºC. Figure 12.5 shows the ladder diagram that might be used. When the temperature rises to become equal to or greater than 100ºC, then the greater than comparison element gives a 1 output and so sets an internal relay. There is then an output. This output latches the greater than comparison element and so the output remains on, even when the temperature falls below 100ºC. The output is not switched off until the less than 90ºC gives an output and resets the internal relay.

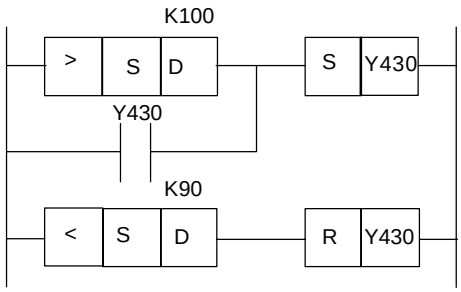


Figure 12.5 *Alarm program*

Another example of the use of comparison is when, say, four outputs need to be started in sequence, i.e. output 1 starts when the initial switch is closed, followed some time later by output 2, some time later by output 3 and some time later by output 4. While this could be done using three timers, another possibility is to use one timer with greater than or equal elements. Figure 12.6 shows a possible ladder diagram.

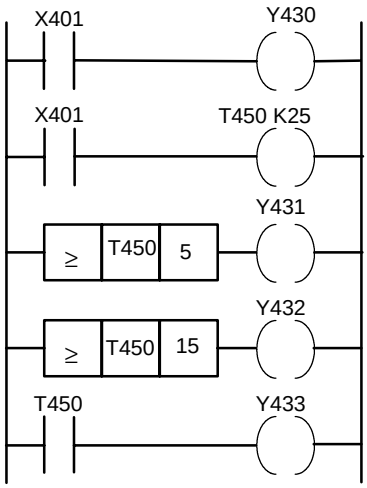


Figure 12.6 *Sequential switching on*

When the X401 contacts close the output Y430 starts. Also the timer is started. When the timer accumulated value reaches 5 s then the greater than or equal to element switches on Y431. When the timer accumulated value reaches 15 s then the greater than or equal to element switches on Y432. When the timer reaches 25 s then its contacts switch on Y433.

12.3 Arithmetic functions

Most PLCs provide BCD-to-binary and binary-to-BCD conversions for use when the input might be for a thumbwheel switch or the output to a decimal display. Figure 12.7(a) shows the typical form of instructions for use in such situations and Figure 12.7(b) the form used by Siemens.

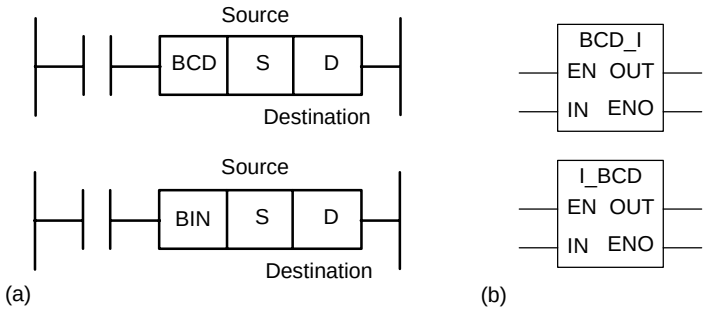


Figure 12.7 *Conversion: (a) BCD-to-binary and binary-to-BCD, (b) BCD-to-integer and integer-to-BCD*

12.3.1 Arithmetic operations

Some PLCs are equipped to carry out just the arithmetic operations of addition and subtraction, others the four basic arithmetic operations of addition, subtraction, multiplication and division, while others can carry out these and various other functions such as the exponential. Addition and subtraction operations are used to alter the value of data held in data registers. For example, this might be to adjust a sensor input reading or perhaps obtain a value by subtracting two sensor values or alter the preset values used by timers and counters. Multiplication might be used to multiply some input before perhaps adding to or subtracting it from another.

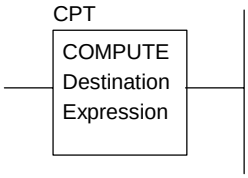


Figure 12.8 *Allen-Bradley format*

The way in which PLCs have to be programmed to carry out such operations varies. Allen-Bradley in some of their PLCs use a compute (CPT) instruction. This is an output instruction that performs the operations defined and then writes the results to a specified destination address. Figure 12.8 shows the instruction format. When the compute instruction is on the programming screen, the destination has to be first entered, then the expression. Thus we might have a destination of T4:1.ACC and an expression (N7:1 + N10:1)*3.5. Note that the symbol * is used for multiplication. The expression states that the value in N7:1 is to be added to the value in N10:1. This sum is then to be multiplied by 3.5. The result is then to be sent to the destination which is the accumulated value in timer 4.1.

Figure 12.9 shows the basic form of the Siemens instuctions for arithmetic functions. With integers the functions avaialble are ADD_1 for addition, SUB_1 for subtraction, MUL_1 for multiplication and DIV_1 for division with the quotient as the result. The arithmetic functions are executed of there is a 1 at the enable EN input.

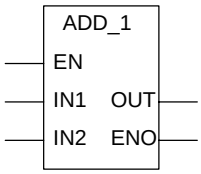


Figure 12.9 *Siemens format*

12.4 Closed loop control

You can control the temperature of a room by switching on an electric fire. The fire will heat the room up to the maximum temperature that is possible bearing in mind the rate at which the fire heats the room and the rate at which is loses heat. This is termed open loop control in that there is no feedback to the fire to modify the rate at which it is heating the room. To do this you need a thermostat which can be set to switch the fire on when the room temperature is below the required value and switch it off when it goes above it. There is feedback of temperature information in this system and as such it it termed closed loop control.

Closed loop control of some variable, e.g. the control of the temperature in a room, is achieved by comparing the actual value for the variable with the desired set value and then giving an output, such as switching on a heater, to reduce the difference. Figure 12.10 illustrates this by means of a block diagram. The actual value of the variable is compared with the set value and a signal obtained representing the difference or error. A controller then takes this difference signal and gives an output to an actuator to give a response to correct the discrepancy. Such a system is called *closed-loop control*.

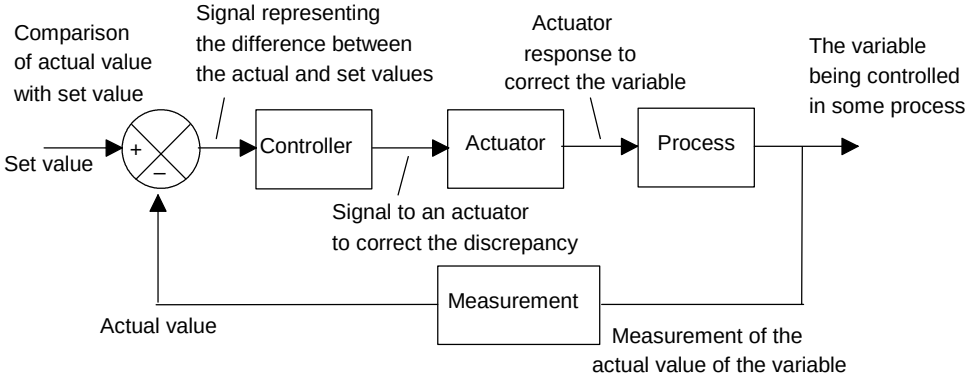


Figure 12.10 *Closed loop control*

Figure 12.11 shows the arrangement that might be used with a PLC used to exercise the closed-loop control. It has been assumed that the actuator and the measured values are analogue and thus require conversion to digital; analogue-to-digital and digital-to-analogue units have thus been shown.

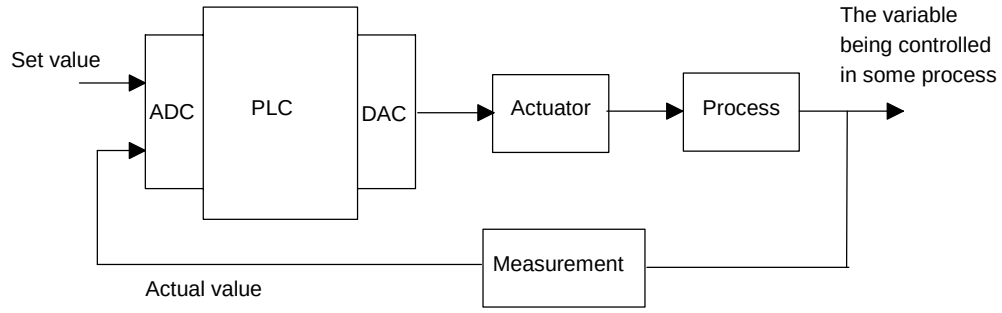


Figure 12.11 *PLC for closed-loop control*

12.4.1 Modes of control

With *proportional control* the controller gives an output to the actuator which is proportional to the difference between the actual value and the set value of the variable, i.e. the error (Figure 12.12(a)). Such a form of control can be given by a PLC with basic arithmetic facilities. The set value and the actual values are likely to be analogue and so are converted to digital and then the actual value is subtracted from the set value and the difference multiplied by some constant, the proportional constant K_P, to give the output, which after conversion to analogue is the correction signal applied to the actuator:

$$\text{controller output} = K_P \times \text{error}$$

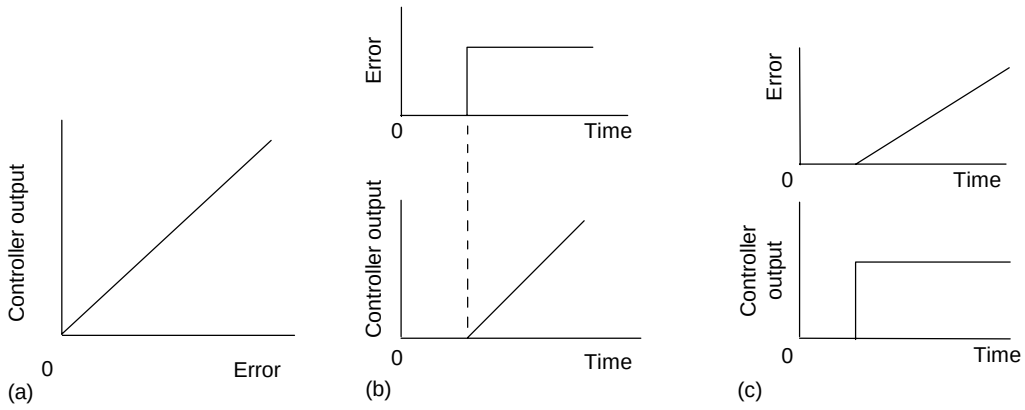


Figure 12.12 *Control: (a) proportional, (b) integral, (c) derivative*

Proportional control has a disadvantage in that, because of time lags inherent in the system, the correcting signal applied to the actuator tends to cause the variable to oscillate about the set value. What is needed is a correcting signal which is reduced as the variable gets close to the set value. This is obtained by *PID control*, the controller giving a correction signal which is computed from a proportional element, the P term, an element which is related to previous values of the variable, the integral I term, and an element related to the rate at which the variable is changing, the derivative D term. With integral control the controller output is proportional to the integral of the error with time, i.e. the area under the error-time graph (Figure 12.12(b)).

controller output = $K_I \times$ integral of error with time

With derivative control the controller output is proportional to the rate at which the error is changing, i.e. the slope of the error-time graph (Figure 12.12(c)):

controller output = $K_D \times$ rate of change of error

The term *tuning* is used for determining the optimum values of K_P, K_I and K_D to be used for a particular control system. The value of K_D/K_P is called the derivative action time T_D, the value of K_P/K_I the integral action time T_I and it is these terms K_P, T_D and T_I which are generally specified.

12.4.2 PID control with a PLC

Many PLCs provide the PID calculation to determine the controller output as a standard routine. All that is then necessary is to pass the desired parameters, i.e. the values of K_P, K_I and K_D, and input/output locations to the routine via the PLC program. Figure 12.13 shows the IEC 1131-3 standard symbol for the PID control function. When AUTO is set the function blocks calculate the output value XOUT needed to bring the variable closer to the required set value.

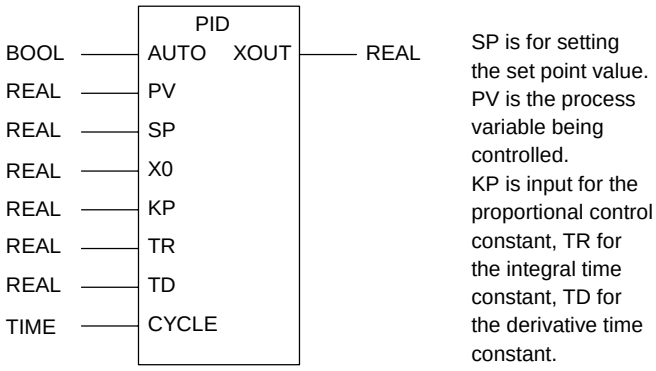


Figure 12.13 *IEC 1131-3 standard symbol*

Problems

Questions 1 to 9 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

Problems 1 and 2 refer to Figure 12.14 which shows two formats used for the move operation.

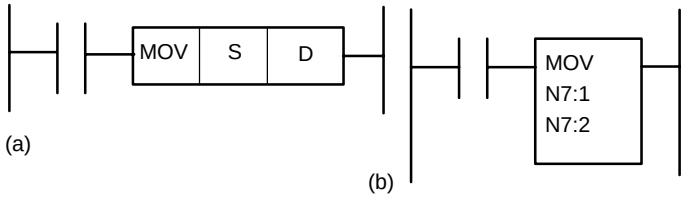


Figure 12.14 *Problems 1 and 2*

1 Decide whether each of these statements is True (T) or False (F).

In Figure 12.14(a), the program instruction is to:
(i) Move the value in S to D, leaving S empty.
(ii) Copy the value in S and put it in D.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

2 Decide whether each of these statements is True (T) or False (F).

In Figure 12.14(b), the program instruction is to:
(i) Move the value in N7:1 to N7:2, leaving N7:1 empty.
(ii) Copy the value in N7:1 and put it in N7:2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

Problems 3 and 4 refer to Figure 12.15 which shows two versions of a ladder rung involving a comparison.

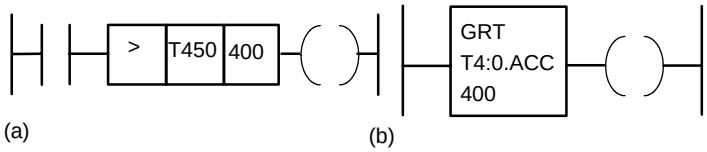


Figure 12.15 *Problems 3 and 4*

3 Decide whether each of these statements is True (T) or False (F).

In Figure 12.15(a), the program instruction is to give an output:
(i) When the accumulated time in timer T450 exceeds a value of 400.
(ii) Until the accumulated time in timer T450 reaches a value of 400.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

4 Decide whether each of these statements is True (T) or False (F).

In Figure 12.15(b), the program instruction is to give an output:
(i) When the accumulated time in timer T4:0 exceeds a value of 400.
(ii) Until the accumulated time in timer T4:0 reaches a value of 400.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

5 Decide whether each of these statements is True (T) or False (F).

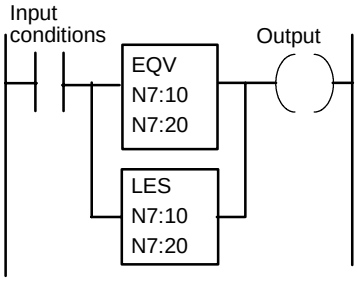


Figure 12.16 *Problem 5*

In Figure 12.16 the program instruction is, when the input conditions are met, to give an output when the data:
(i) In N7:10 equals that in N7:20.
(ii) In N7:10 is less than that in N7:20.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

6 Decide whether each of these statements is True (T) or False (F).

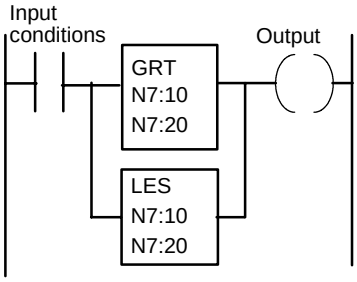


Figure 12.17 *Problem 6*

In Figure 12.17, the program instruction is to give, when the input conditions are met, an output when:
(i) The data in N7:10 is not equal to that in N7:20.
(ii) The data in N7:10 is greater or less than that in N7:20.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

7 In Figure 12.18, when the input conditions are met, the program instruction is to give in N7:20:

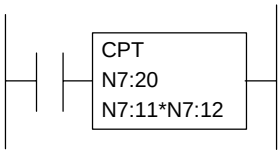


Figure 12.18 *Problem 7*

A The sum of the data in N7:11 and N7:12.
B The product of the data in N7:11 and N7:12.
C The difference between the data in N7:11 and N7:12.
D The value given by dividing the data in N7:11 by that in N7:12.

8 Decide whether each of these statements is True (T) or False (F).

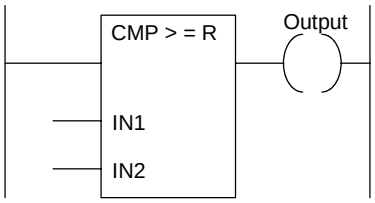


Figure 12.19 *Problem 8*

For the Siemens function box shown in Figure 12.19, the output will be set when:
(i) Inputs IN1 and IN2 are both the same REAL number.
(ii) Input IN1 is a REAL number greater than input IN2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

9 Decide whether each of these statements is True (T) or False (F).

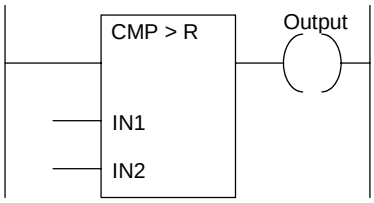


Figure 12.20 *Problem 9*

For the Siemens function box shown in Figure 12.20, the output will be be set when:
(i) Inputs IN1 and IN2 are both the same REAL number.
(ii) Input IN1 is a REAL number greater than input IN2.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

10 Devise ladder programs for systems that will carry out the following tasks:
(a) Switch on a pump when the water level in a tank rises above 1.2 m and switch it off when it falls below 1.0 m.
(b) Switch on a pump, then 100 s later switch on a heater, then a further 30 s later switch on the circulating motor.

13 Designing systems

This chapter considers how programs are designed and how they and a PLC system can be tested and faults found. This involves consideration of both the hardware and the software.

13.1 Program development

Whatever the language in which a program is to be written, a systematic approach to the problem can improve the chance of high quality programs being generated in as short a time as possible. A systematic design technique is likely to involve the following steps:

1 A definition of what is required with the inputs and outputs specified.
2 A definition of the algorithm to be used. An algorithm is a step-by-step sequence which defines a method of solving the problem. This can often be shown by a flow chart or written in pseudocode, this involving the use of the words BEGIN, DO, END, IF-THEN-ELSE, WHILE-DO.
3 The algorithm is then translated into instructions that can be inputted to the PLC. Because programs are often very long and can end up difficult to write as a long single block and even more difficult to later follow for fault finding and maintenance, it is advisable to break the program down into areas which are then further subdivided until manageable size blocks of program occur. This is termed *top-down design*.
4 The program is then tested and debugged.
5 The program is documented so that any person using or having to modify the program at a later date understands how the program works.

13.1.1 Flow charts and pseudocode

Figure 13.1(a) shows the symbols used in flow charts. Each step of an algorithm is represented by one or more of these symbols and linked by lines to represent the program flow, Figure 13.1(b) illustrating this. Pseudocode is a way of describing the steps in an algorithm in an informal way.

Consider how the following program operations can be represented by flow charts and pseudocode and then programmed using ladder and sequential function chart programming:

1 *Sequential*
Consider a sequence when event A has to be followed by event B. Figure 13.2(a) shows how this can be represented by a flow chart.

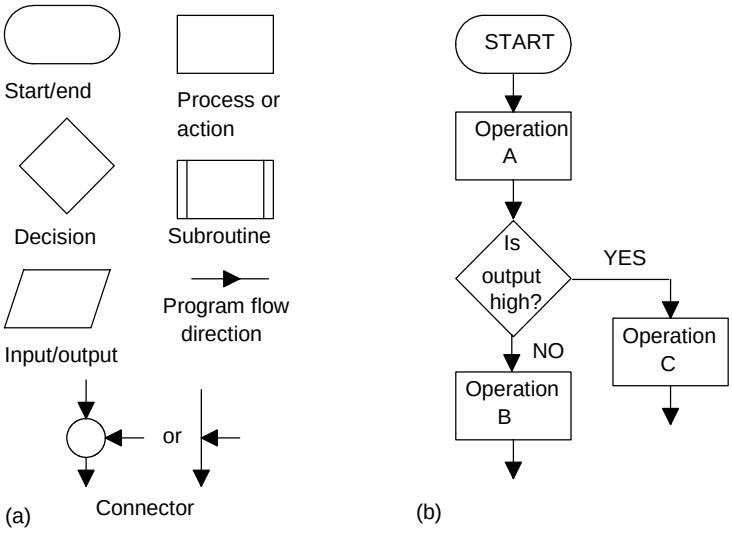


Figure 13.1 *(a) Flow chart symbols, (b) example of a simple flow chart*

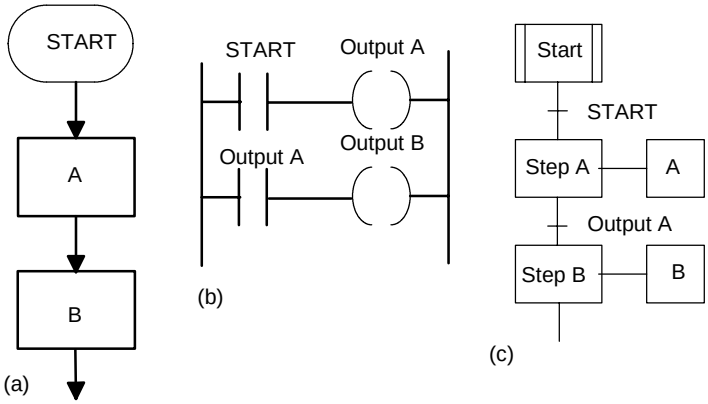


Figure 13.2 *Sequence*

In pseudocode this is written as:

```
BEGIN A
    DO A
END A
BEGIN B
    DO B
END B
```

A sequence can be translated into a ladder program in the way shown in Figure 13.2(b). When the start input occurs then output A happens. When action A happens it operates output A relay and results in output B occurring. Figure 13.2(c) shows the sequential function chart representation of a sequence.

2 *Conditional*

Figure 13.3(a) shows the flow chart for when A or B is to happen if a particular condition X being YES or NO occurs. The pseudocode to describe this involves the words IF-THEN-ELSE-ENDIF.

```
IF X
THEN
    BEGIN A
    DO A
    END A
ELSE
    BEGIN B
    DO B
    END B
ENDIF X
```

Such a condition can be represented by the ladder diagram shown in Figure 13.3(b). When the start input occurs, the output will be A if there is an input to X, otherwise the output is B. Figure 13.3(c) shows the sequential function chart for such selective branching.

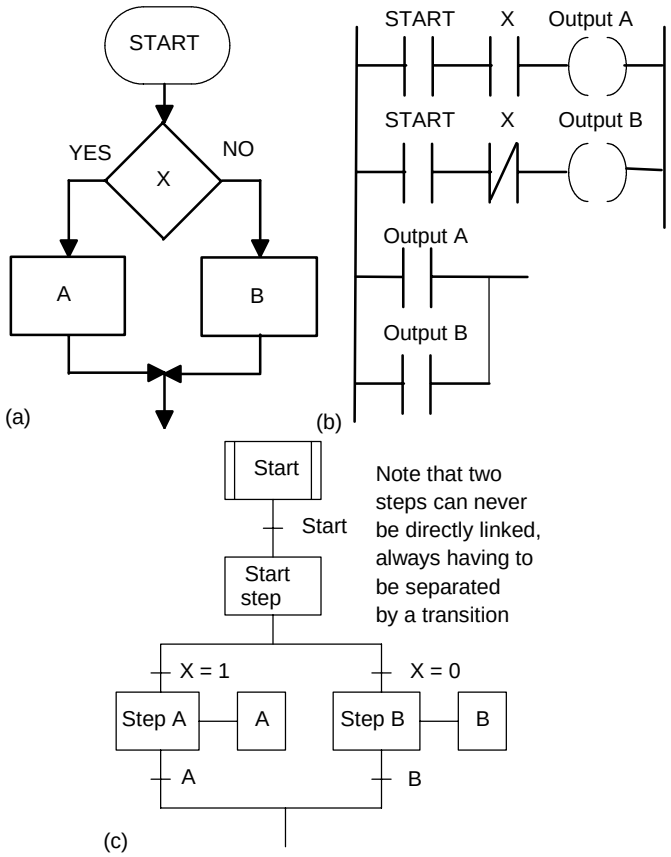
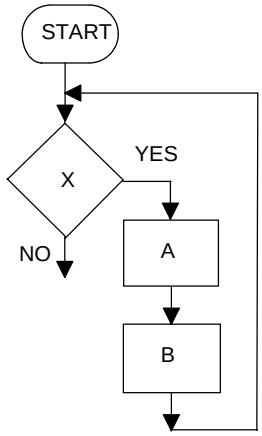


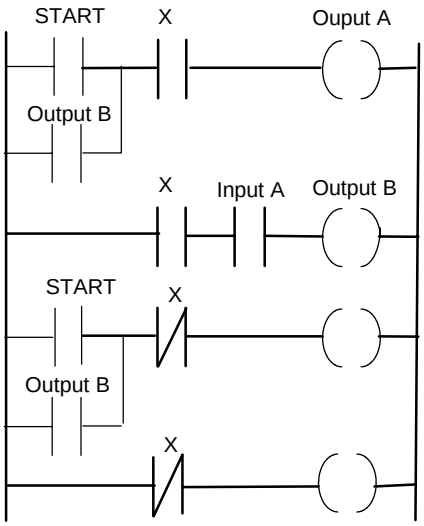
Figure 13.3 *Conditional*

3 *Looping*

A loop is a repetition of some element of a program, the element being repeated as long as some condition prevails. Figure 13.4(a) shows how this can be represented by a flow chart. As long as condition X is realised then the sequence A followed by B occurs and is repeated. When X is no longer realised then the program continues and the looping through A and B ceases.



(a)

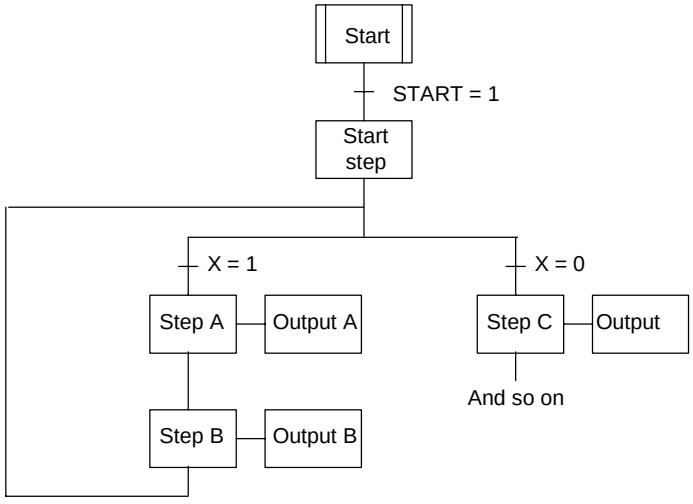


While X yes output A, which when completed gives input A as yes and so output B.

When B happens it latches the START. and so the decision X is faced again.

Rest of program occurs while not X

(b) Rest of program



(c)

Figure 13.4 *Looping*

In pseudocode this can be represented by using the words WHILE-DO -ENDWHILE:

WHILE X

```
        BEGIN A
        DO A
        END A
        BEGIN B
        DO B
        END B
    ENDWHILE X
```

Figure 13.4(b) shows how this can be represented by a ladder diagram and using an internal relay. Figure 13.4(c) shows the sequential flow chart.

Where a loop has to be repeated for a particular number of times, a counter can be used, receiving an input pulse each time a loop occurs and switching out of the loop sequence when the required number of loops has been completed (Figure 13.5).

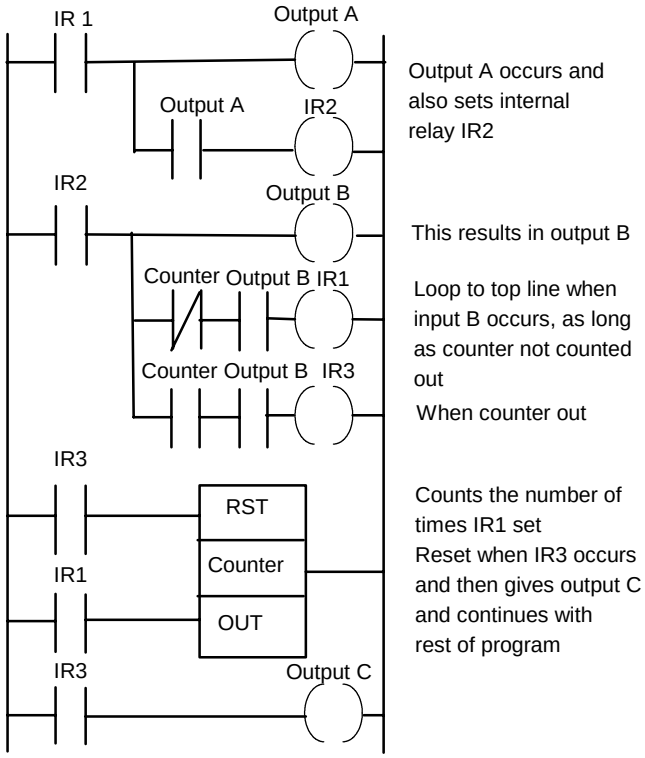


Figure 13.5 *Looping*

13.2 Safe systems

Modern safety legislation charges employers with duties which include making the workplace safe and without risks to health and ensuring that plant and machinery are safe and that safe systems of work are set and followed. There is thus a need to assess the risks in the workplace. This means looking for hazards, i.e. anything that can cause harm, deciding who might be harmed and how, evaluating the risks that somebody will be harmed by a hazard and whether existing precautions are adequate or

whether more needs to be done to reduce the chance of harm occurring, recording the findings and reviewing and revising the assessment if necessary.

Thus, for example, issues such as emergency stops and access doors on equipment need to be considered, the risks assessed and safe systems then designed. With regard to access doors on equipment, switch contacts can be used on such doors so that the system is stopped if the doors are not correctly guarding equipment.

An important standard is IEC (International Electrotechnical Commission) 61508: Functional safety of electrical/electronic/ programmable electronic safety-related systems. The standard is in seven parts, namely Part 1: General requirements; Part 2: Requirements for E/E/PE safety-related systems; Part 3: Software requirements; Part 4: Definitions and abbreviations; Part 5: Examples of methods for the determination of safety integrity levels; Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3; Part 7: Overview of techniques and measures. In order to provide functional safety of a machine or plant, the safety-related protective or control system must function correctly and when a failure occurs it must operate so that the plant or machine is brought into a safe shut-down state.

13.2.1 PLC systems and safety

Safety must be a priority in the design of a PLC system. Thus, emergency stop buttons and safety guard switches must be hard wired and not depend on the PLC software for implementation so that, in the situation where there is a failure of the stop switch or PLC, the system is automatically safe. The system must be *fail-safe*.

With a PLC system, a stop signal can be provided by a switch as shown in Figure 13.6. This arrangement is unsafe as an emergency stop because if there is a fault and the switch cannot be operated, then no stop signal can be provided. Thus to start we momentarily close the press-button start switch and the motor control internal relay then latches this closure and the output remains on. To stop we have to momentarily open the stop switch; this unlatches the start switch. However, if the stop switch cannot be operated then we cannot stop the system. What we require is a system that will still stop if a failure occurs in the stop switch.

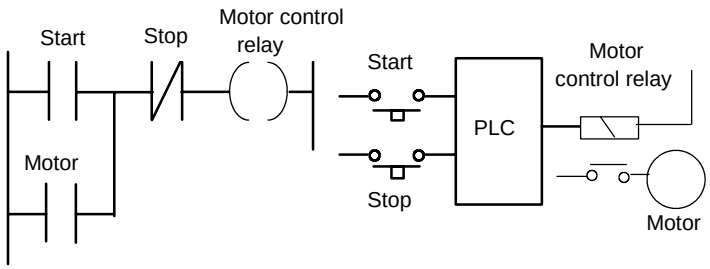


Figure 13.6 *An unsafe stop system*

We can achieve this by the arrangement shown in Figure 13.7. The program has the stop switch as open contacts. However, because the

hardwired stop switch has normally closed contacts then the program has the signal to close the program contacts. Pressing the stop switch opens the program contacts and stops the system.

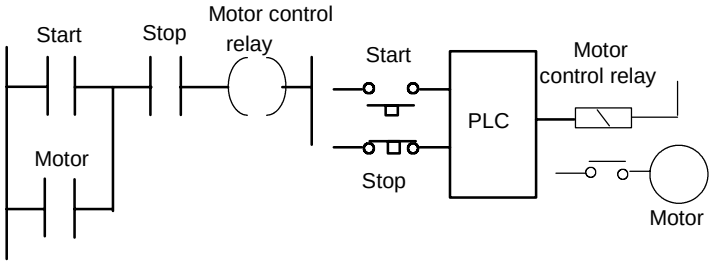


Figure 13.7 *A safer stop system*

For a safe emergency stop system we need one which will provide a stop signal if there is a fault and the switch cannot be operated. Because there might be problems with a PLC, we also need the emergency stop to operate independently of the PLC. Putting the emergency stop in the input to the PLC gives an unsafe system (Figure 13.8).

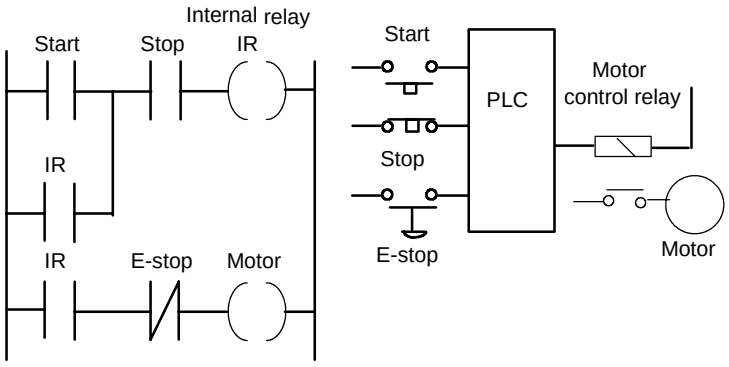


Figure 13.8 *An unsafe emergency stop system*

Figure 13.9 shows a safer system where the emergency stop switch is hard-wired in the output. Pressing the emergency stop button switch stops, say, a motor running. On releasing the stop button, the motor will not restart again because the internal relay contacts have become unlatched.

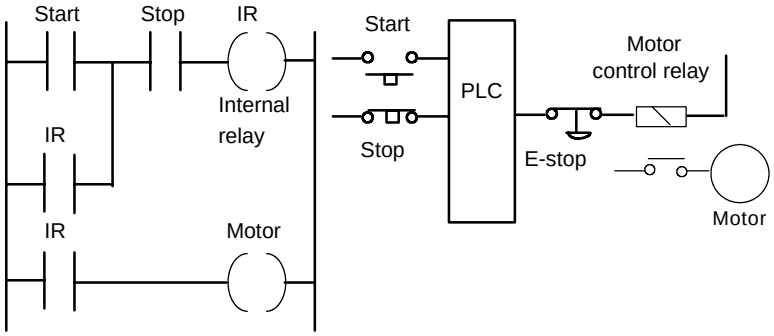


Figure 13.9 *A safer emergency stop system*

13.2.2 Emergency stop relays

Emergency stop relays are widely used for emergency stop arrangements, e.g. the PNOZ p1p from Pilz GmbH & Co. This has LEDs for indicating the status of input and output circuits, the reset circuit and power supply and faults. However, the base unit can be connected via an interface module so that its status can be read by a PLC. This interface isolates the output from the emergency stop relay from the signal conditioning and input to the PLC by means of optoisolators (see Figure 1.8). Thus, while the emergency stop operates independently of the PLC it can provide signals which a PLC can use to, say, initiate safe closing-down procedures. Figure 13.10 illustrates this.

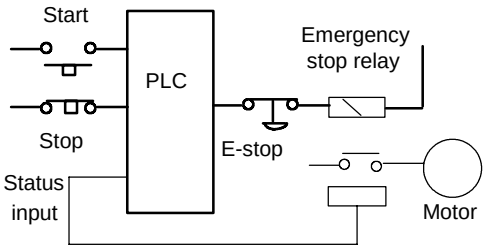


Figure 13.10 *Emergency stop relay providing feedback of status*

A simple emergency stop relay in which operation of the emergency-stop button breaks the control circuit to the relay, causing it to de-energise and switch off the power (Figure 13.11(a)), has the problem that if the relay contacts weld together the emergency stop will not operate. This can be overcome by using a dual channel mode of operation in which there are two normally closed contacts in series and both are broken by the action of the relay de-energising (Figure 13.12(b)). Safety can be increased yet further if three contacts in series are used, one using normally closed contacts and the others normally open contacts. Then one set of contacts has to be de-energised and the other two energised.

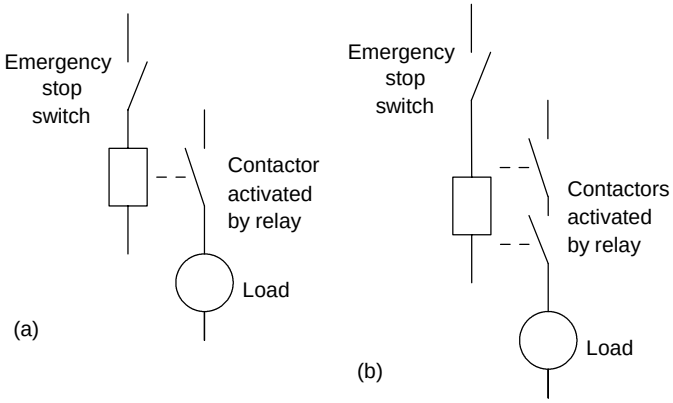


Figure 13.11 *Emergency stop relay: (a) single channel mode, (b) dual channel mode.*

Such types of stop-relays systems are used with so-called 'light curtains'. A danger zone, such as a packaging machine, is protected from people getting too close by infrared beams. If a light beam is broken it immediately triggers a safe shutdown command. Safety mats are another way of detecting when someone is too close to a machine. They are placed round a machine and when someone steps on the mat a contact is closed which causes the machine to stop.

Thus a safe-operating system for a work cell might use gated entry systems, e.g. guards on machines which activate stop relays if they are not in place, light curtains and emergency stop relays.

13.2.3 Safety PLCs

Safety PLCs are specially designed PLCs to enable safety functions to be realised. In a safety PLC there can be two or three microprocessors that perform exactly the same logic, check against each other and only give outputs if there is agreement. An example of such a PLC is the SIMATIC S5-95F. This is a two-channel system with two identical sub-systems which communicate with each other via a fibre-optic cable link. The inputs from the sensors are fed simultaneously to both sub-systems. During operation, data is passed via the fibre-optic cable between the two sub-systems. They operate in synchronism with the same program and compare input and output signals, the results of logic operations, counters, etc. and automatically goes into a safe stop condition if there are different outputs or internal faults or failures. For safety-related digital outputs, actuators are switched on or off from both sub-systems. This means that one sub-system can alone shut down equipment.

13.3 Commissioning

Commissioning of a PLC system involves:

1 Checking that all the cable connections between the PLC and the plant being controlled are complete, safe and to the required specification and meeting local standards.
2 Checking that the incoming power supply matches the voltage setting for which the PLC is set.
3 Checking that all protective devices are set to their appropriate trip settings.
4 Checking that emergency stop buttons work.
5 Checking that all input/output devices are connected to the correct input/output points and giving the correct signals.
6 Loading and testing the software.

13.3.1 Testing inputs and outputs

Input devices, e.g. switches, can be manipulated to give the open and closed contact conditions and the corresponding LED on the input module observed. It should be illuminated when the input is closed and not illuminated when it is open. Failure of an LED to illuminate could be because the input device is not correctly operating, there are incorrect wiring connections to the input module, the input device is not correctly powered or the LED or input module is defective. For output devices that

can be safely started, push buttons might have been installed so that each output can be tested.

Another method that can be used to test inputs and outputs is termed *forcing*. This involves software, rather than mechanical switching on or off, being used with instructions from the programming panel to turn off or on inputs/outputs. In order to do this, a PLC has to be switched into the forcing or monitor mode by perhaps pressing a key marked FORCE or selecting that mode on a screen display. For example, Figure 13.12 shows the keystrokes that might be used and the resulting screen display, to force the output Y005 into the on state. Figure 13.13 shows the keys for the forcing of an input X001 into a closed state. Thus if an input is forced and the input LED comes on then we can check that the consequential action of that input being on occurs.

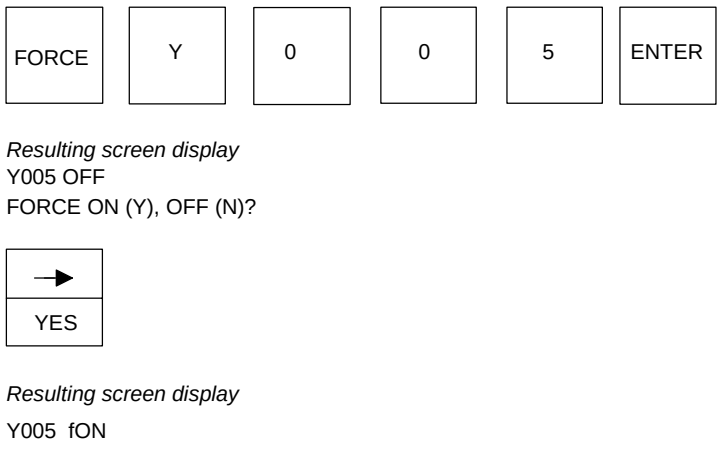


Figure 13.12 *Forcing an output*

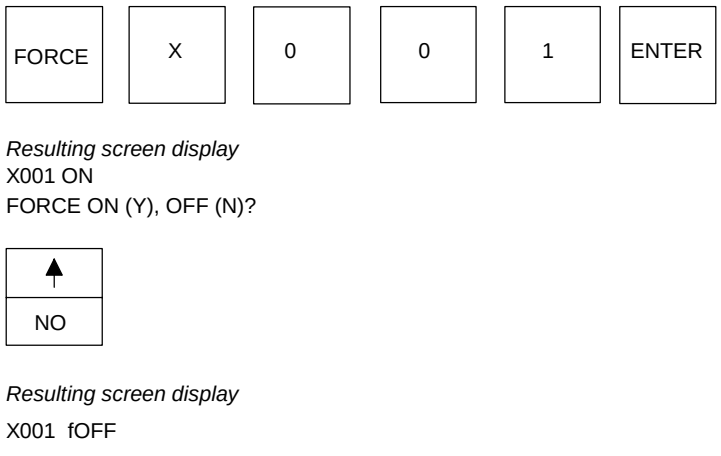


Figure 13.13 *Forcing an input*

13.3.2 Testing software

Most PLCs contain some software checking program. This checks through the installed program for incorrect device addresses, and provides a list on

a screen or as a printout of all the input/output points used, counter and timer settings, etc. with any errors detected. For example, there might be a message for a particular output address that it is used as an output more than once in the program, a timer or counter is being used without a preset value, a counter is being used without a reset, etc.

13.3.3 Simulation

Many PLCs are fitted with a simulation unit which reads and writes information directly into the input/output memory and so simulates the actions of the inputs and outputs. The installed program can thus be run and inputs and outputs simulated so that they, and all preset values, can be checked. To carry out this type of operation the terminal has to be placed in the correct mode. For Mitsubishi this is termed the monitor mode, for Siemens the test mode, for Telemecanique the debug mode.

With Mitsubishi in the monitor mode, Figure 13.14 shows how inputs appear when open and closed, and output when not energised and energised. The display shows a selected part of the ladder program and what happens as the program proceeds. Thus at some stage in a program the screen might appear in the form shown in Figure 13.15(a). For rung 12, with inputs to X400, X401 and X402, but not M100, there is no output from Y430. For rung 13, the timer T450 contacts are closed, the display at the bottom of the screen indicating that there is no time left to run on T450. Because Y430 is not energised the Y430 contacts are open and so there is no output from Y431. If we now force an input to M100 then the screen display changes to that shown in Figure 13.5(b). Now Y430, and consequently Y431, come on.

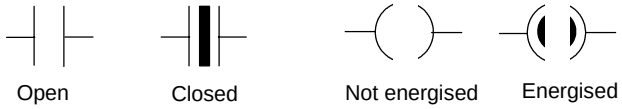


Figure 13.14 *Monitor mode symbols*

13.4 Fault finding

With any PLC controlled plant, by far the greater percentage of the faults are likely to be with sensors, actuators and wiring rather than within the PLC itself. Of the faults within the PLC, most are likely to be in the input/output channels or power supply rather than in the CPU.

As an illustration of a fault, consider a single output device failing to turn on though the output LED is on. If testing of the PLC output voltage indicates that it is normal then the fault might be a wiring fault or a device fault. If checking of the voltage at the device indicates the voltage there is normal then the fault is the device. As another illustration, consider all the inputs failing. This might be as a result of a short circuit or earth fault with an input and a possible procedure to isolate the fault is to disconnect the inputs one by one until the faulty input is isolated. An example of another fault is if the entire system stops. This might be a result of a power failure, or someone switching off the power supply, or a circuit breaker tripping.

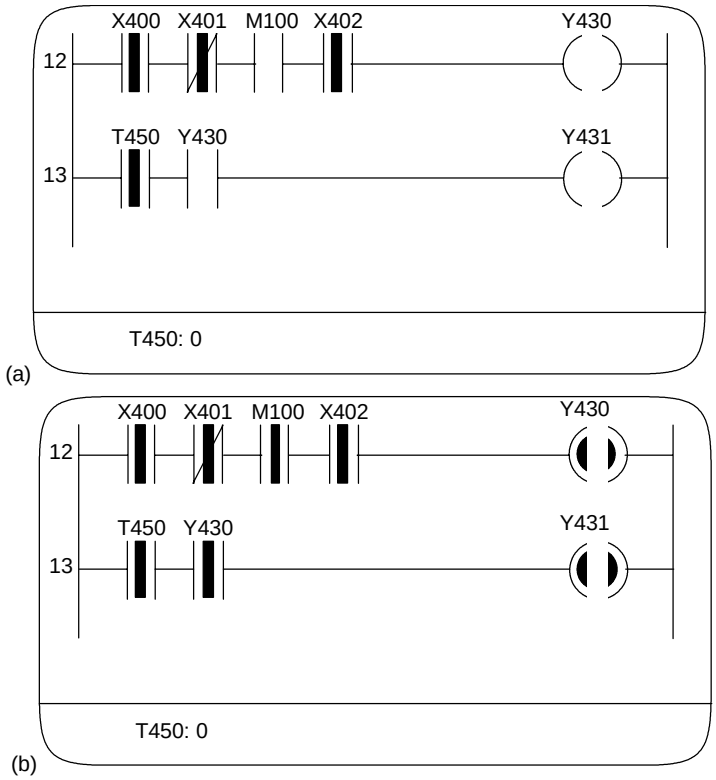


Figure 13.15 *Ladder program monitoring*

Many PLCs provide built-in fault analysis procedures which carry out self-testing and display fault codes, with possibly a brief message, which can be translated by looking up the code in a list to give the source of the fault and possible methods of recovery. For example, the fault code may indicate that the source of the fault is in a particular module with the method of recovery given as replace that module or perhaps switch the power off and then on.

13.4.1 Fault detection techniques

The following are some of the common fault detection techniques used:

1 *Timing checks*
The term *watchdog* is used for a timing check that is carried out by the PLC to check that some function has been carried out within the normal time. If the function is not carried out within the normal time then a fault is assumed to have occurred and the watchdog timer trips, setting off an alarm and perhaps closing down the PLC. As part of the internal diagnostics of PLCs, watchdog timers are used to detect for faults. The watchdog timer is preset to a time slightly longer than the scan time would normally be. They are then set at the beginning of each program scan and if the cycle time is as normal they do not time out and are reset at the end of a cycle ready for the next cycle.

However, if the cycle time is longer than it would normally be, the watchdog timer times out and indicates that the system has a fault.

Within a program additional ladder rungs are often included so that when a function starts a timer is started. If the function is completed before the time runs out then the program continues but if not the program uses the jump command to move to a special set of rungs which triggers off an alarm and perhaps stops the system. Figure 13.16 shows an example of a watchdog timer which might be used with the movement of a piston in a cylinder. When the start switch is closed the solenoid of a valve is energised and causes the piston in the cylinder to start moving. It also starts the timer. When the piston is fully extended it opens a limit switch and stops the timer. If the time taken for the piston to move and switch off the timer is greater than the preset value used for the timer, the timer sets off the alarm.

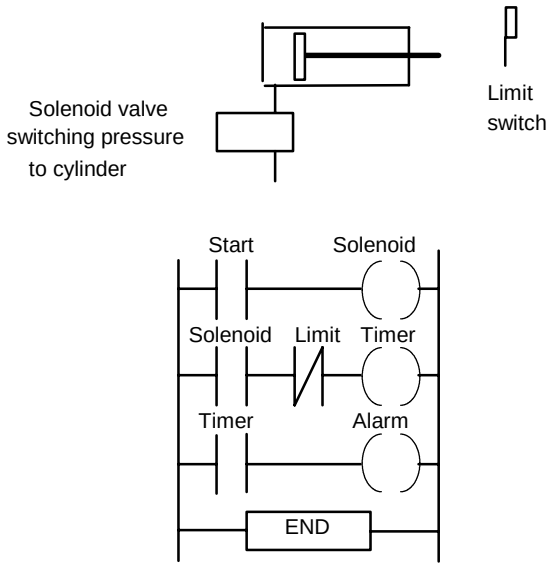


Figure 13.16 *Watchdog timer*

2 *Last output set*

This technique involves the use of status lamps to indicate the last output that has been set during a process which has come to a halt. Such lamps are built into the program so that as each output occurs a lamp comes on. The lamps on thus indicate which outputs are occurring. The program has to be designed to turn off previous status lamps and turn on a new status lamp as each new output is turned on. Figure 13.17 illustrates this.

Such a system can be cumbersome with a large system where there are many outputs. In such a case, the outputs might be grouped into sets and a status lamp used for each set. A selector switch can then be used within a group to select each output in turn to determine whether it is on. Figure 13.18 illustrates this.

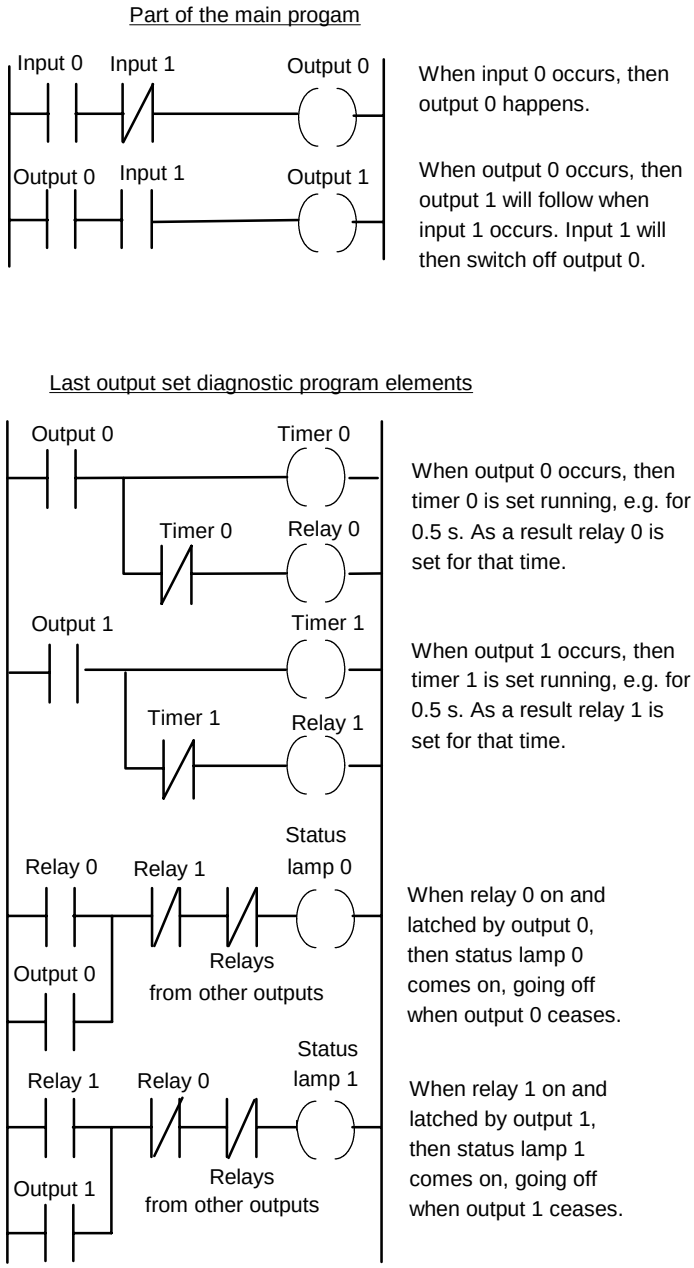

Part of the main progam
Input 0
Input 1
Output 0
When input 0 occurs, then output 0 happens.
Output 0
Input 1
Output 1
When output 0 occurs, then output 1 will follow when input 1 occurs. Input 1 will then switch off output 0.
Last output set diagnostic program elements
Output 0
Timer 0
Timer 0
Relay 0
When output 0 occurs, then timer 0 is set running, e.g. for 0.5 s. As a result relay 0 is set for that time.
Output 1
Timer 1
Timer 1
Relay 1
When output 1 occurs, then timer 1 is set running, e.g. for 0.5 s. As a result relay 1 is set for that time.
Relay 0
Relay 1
Status lamp 0
Relays from other outputs
Output 0
When relay 0 on and latched by output 0, then status lamp 0 comes on, going off when output 0 ceases.
Relay 1
Relay 0
Status lamp 1
Relays from other outputs
Output 1
When relay 1 on and latched by output 1, then status lamp 1 comes on, going off when output 1 ceases.

Figure 13.17 *Last output set diagnostic program*

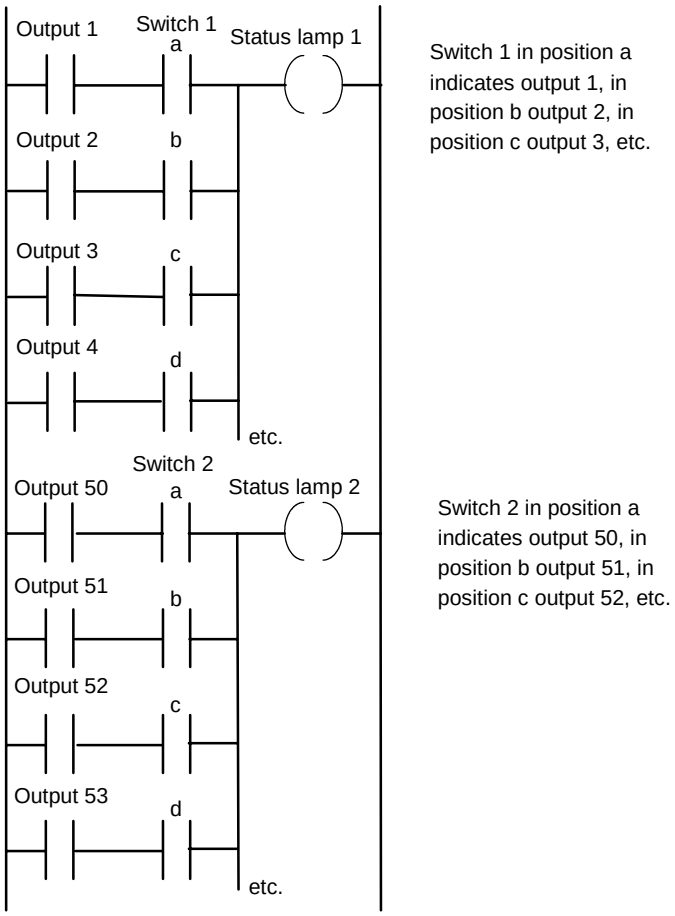


Figure 13.18 *Single status lamp for a group of outputs*

As an illustration of the use of the program to indicate which action occurred last, Figure 13.19 shows the program that might be used with a pneumatic system operating cylinders in a sequence. The program indicates at which point in the sequence a fault occurred, e.g. a piston sticking, and would be added to the main program used to sequence the cylinders. Each of the cylinder movements has a light-emitting diode associated with it, the last cylinder movement being indicated by its LED being illuminated.

3 *Replication*

Where there is concern regarding safety in the case of a fault developing, checks may be constantly used to detect faults. One technique is *replication checks* which involves duplicating, i.e. replicating, the PLC system. This could mean that the system repeats every operation twice and if it gets the same result it is assumed there is no fault. This procedure can detect transient faults. A more expensive alternative is to have duplicate PLC systems and compare the results given by the two systems. In the absence of a fault the two results should be the same, a fault showing up as a difference.

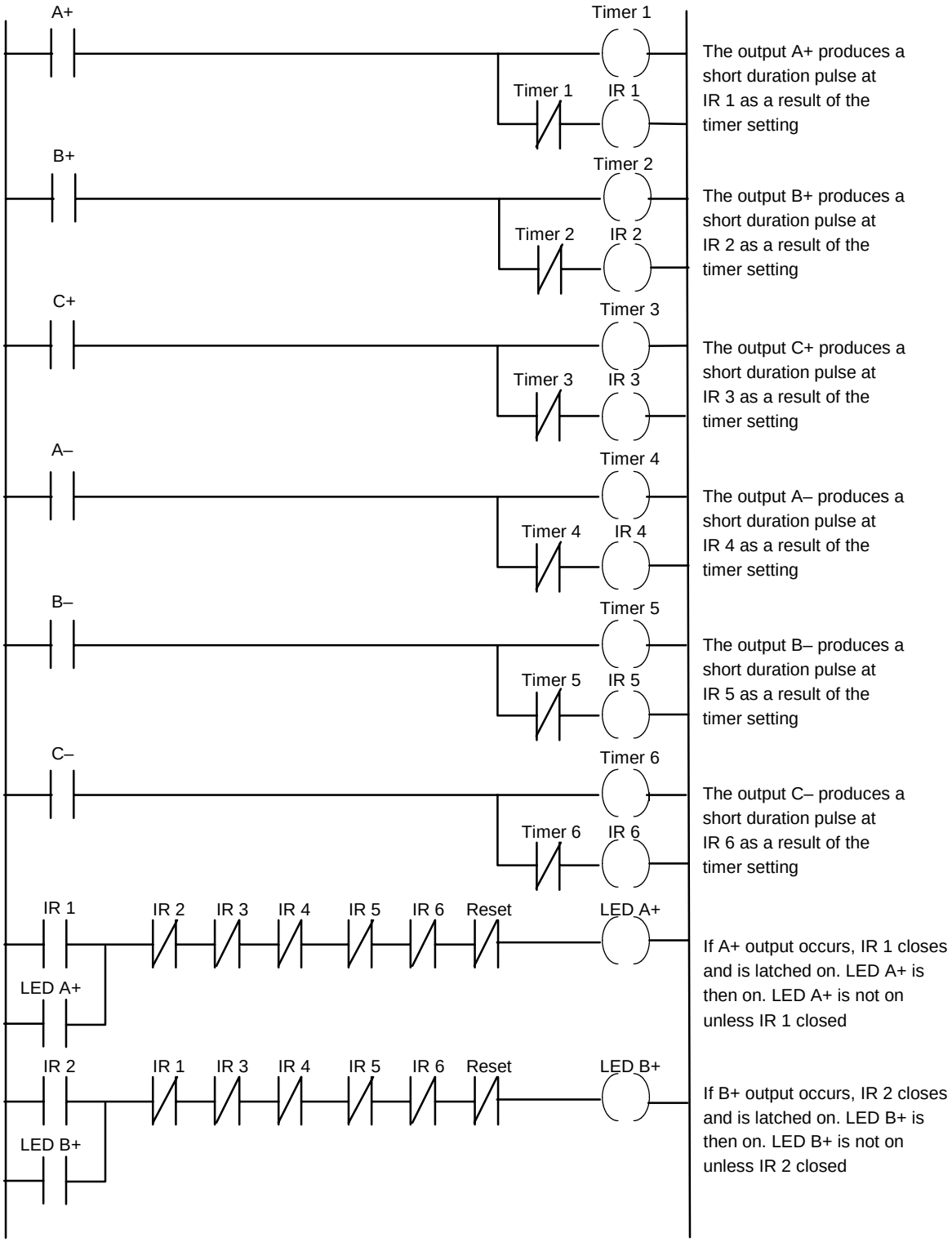

A+
Timer 1
Timer 1
IR 1
The output A+ produces a short duration pulse at IR 1 as a result of the timer setting
B+
Timer 2
Timer 2
IR 2
The output B+ produces a short duration pulse at IR 2 as a result of the timer setting
C+
Timer 3
Timer 3
IR 3
The output C+ produces a short duration pulse at IR 3 as a result of the timer setting
A–
Timer 4
Timer 4
IR 4
The output A– produces a short duration pulse at IR 4 as a result of the timer setting
B–
Timer 5
Timer 5
IR 5
The output B– produces a short duration pulse at IR 5 as a result of the timer setting
C–
Timer 6
Timer 6
IR 6
The output C– produces a short duration pulse at IR 6 as a result of the timer setting
IR 1
IR 2
IR 3
IR 4
IR 5
IR 6
Reset
LED A+
LED A+
If A+ output occurs, IR 1 closes and is latched on. LED A+ is then on. LED A+ is not on unless IR 1 closed
IR 2
IR 1
IR 3
IR 4
IR 5
IR 6
Reset
LED B+
LED B+
If B+ output occurs, IR 2 closes and is latched on. LED B+ is then on. LED B+ is not on unless IR 2 closed

Figure 13.19 *(Continued on next page)*

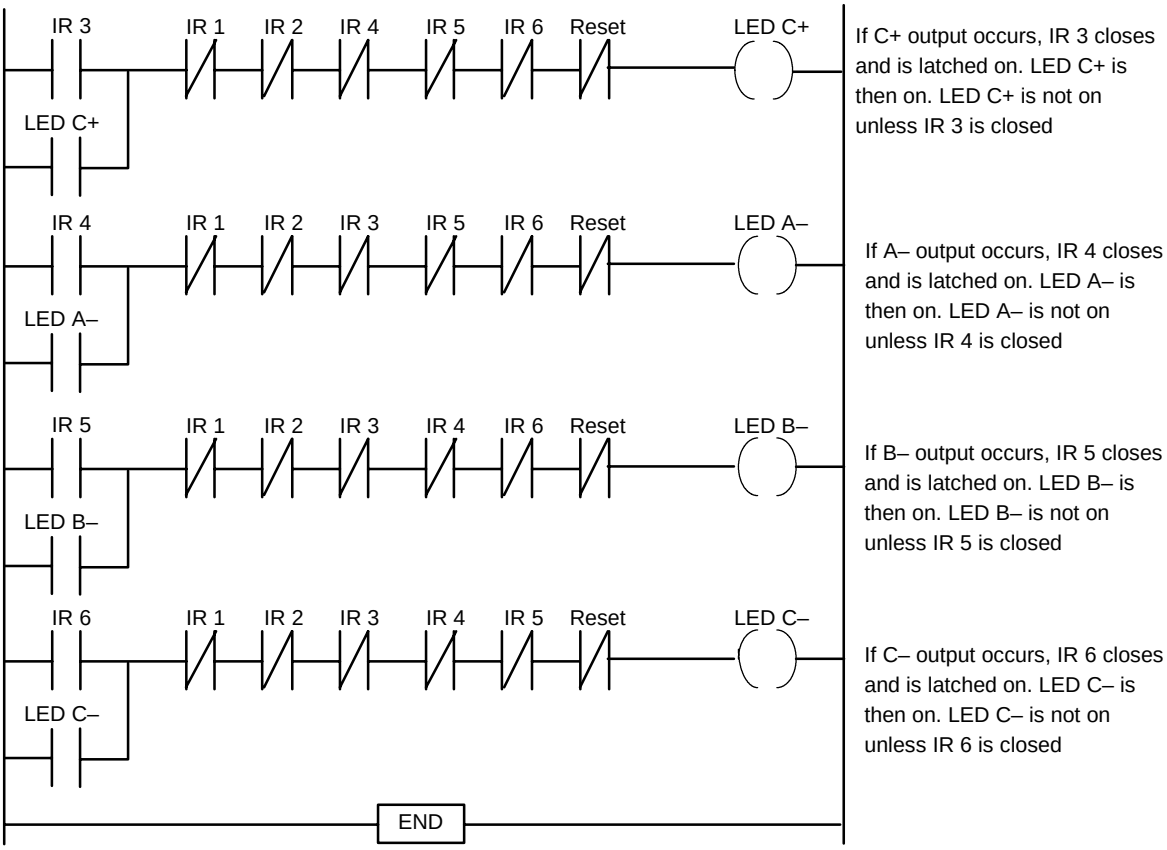

IR 3
IR 1
IR 2
IR 4
IR 5
IR 6
Reset
LED C+
LED C+
If C+ output occurs, IR 3 closes and is latched on. LED C+ is then on. LED C+ is not on unless IR 3 is closed
IR 4
IR 1
IR 2
IR 3
IR 5
IR 6
Reset
LED A–
LED A–
If A– output occurs, IR 4 closes and is latched on. LED A– is then on. LED A– is not on unless IR 4 is closed
IR 5
IR 1
IR 2
IR 3
IR 4
IR 6
Reset
LED B–
LED B–
If B– output occurs, IR 5 closes and is latched on. LED B– is then on. LED B– is not on unless IR 5 is closed
IR 6
IR 1
IR 2
IR 3
IR 4
IR 5
Reset
LED C–
LED C–
If C– output occurs, IR 6 closes and is latched on. LED C– is then on. LED C– is not on unless IR 6 is closed
END

Figure 13.19 *Diagnostic program for last cylinder action*

4 *Expected value checks*
Software errors can be detected by checking whether an expected value is obtained when a specific input occurs. If the expected value is not obtained then a fault is assumed to be occurring.

13.4.2 Program storage

Applications programs may be loaded into battery-backed-up RAM in a PLC. A failure of the battery supply means a complete loss of the stored programs. An alternative to storing applications programs in battery-backed RAM is to use EPROM. This form of memory is secure against the loss of power. Against the possibility of memory failure occurring in the PLC and loss of the stored application program, a back-up copy of each application program should be kept. If the program has been developed using a computer, the back-up may be on a CD or a hard disk. Otherwise the back-up may be on an EPROM cartridge. The program can then again be downloaded into the PLC without it having to be rewritten.

13.5 System documentation

The documentation is the main guide used by the everyday users and for troubleshooting and fault finding with PLCs. It thus needs to be complete and in a form that is easy to follow. The documentation for a PLC installation should include:

1 A description of the plant.
2 Specification of the control requirements.
3 Details of the programmable logic controller.
4 Electrical installation diagrams.
5 Lists of all input and output connections.
6 Application program with full commentary on what it is achieving.
7 Software back-ups.
8 Operating manual, including details of all start up and shut down procedures and alarms.

13.5.1 Example of an industrial program

The following is an example of a how a program might appear for a real plant controlled by an Allen-Bradley PLC5 and I am grateful to Andrew Parr for supplying it. It illustrates the way a program file is documented to aid in clarification and the safety and fault indication procedures that are used. Note that the right-hand power rail has been omitted, this being allowable in IEC 1131-3.

The program is one of about forty program files in the complete programme, each file controlling one area of operation and separated by a page break from the next file. The file that follows controls a bundle cutting band saw and involves motor controls, desk lamps and a small state transition sequence.

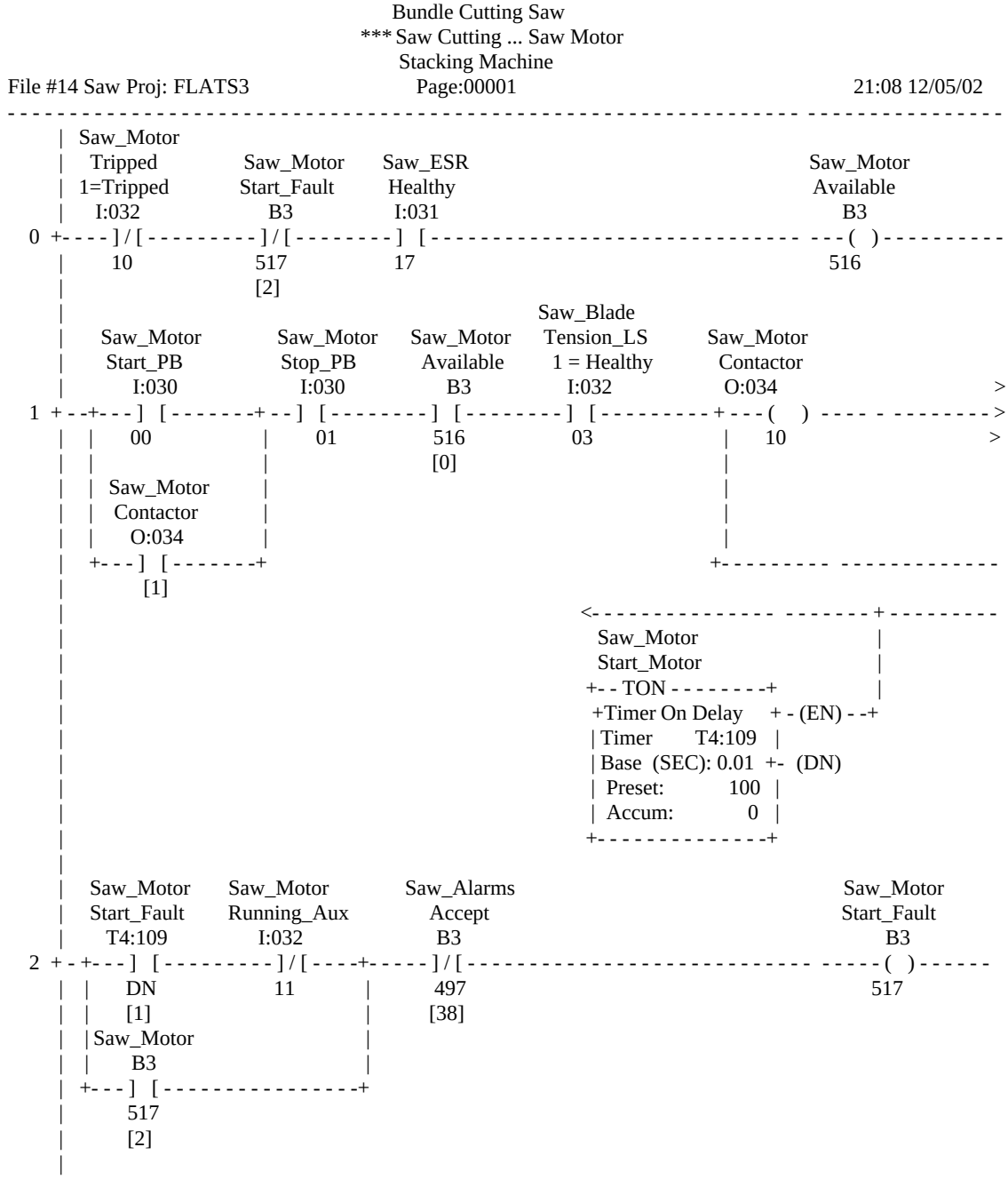
Note the rung cross references, e.g. [38] below B3/497 in rung 2. This is used to show the B3/497 originates in rung 38 in the current program file. Also note that all instructions are tagged with descriptions and the file is broken down into page section. The software allows you to go straight to a function via the page titles.

All the motor starter rungs work in the same way. The PLC energises the contactor and then one second later looks for the auxiliary relay (labelled as Aux in the program file) coming back to say the contactor has energised. If there is a fault which causes the contactor to de-energise, e.g. a loss of supply, a trip or open circuit coil, then it causes the PLC to signal a fault and de-energise the contactor output so that the machine does not spring into life when the fault is cleared.

The saw normally sits raised clear of the bundle. To cut the bundle, the blade motor has to be started and the lower push-button pressed (at rung 8). The saw falls under gravity at a fast or slow speed which is set by hydraulic valves. To raise the saw a hydraulic pump is started to pump oil into the saw support cylinders. At any time the saw can be raised, e.g. to clear swarf, to what is termed the pause state. Otherwise, cutting continues until the bottom limit is reached. The saw then is raised to the top limit for the next bundle. A cut can be aborted by pressing the raise button for two seconds. While a bundle is being cut, it is held by clamp solenoids.

The final three rungs of the program set the length to be cut. There are two photocells about 20 mm apart on a moveable carriage. These are

positioned at the required length. The operator runs the bundle in until the first is blocked and the second is clear. These control the Long/ Correct/Short desk lamps.

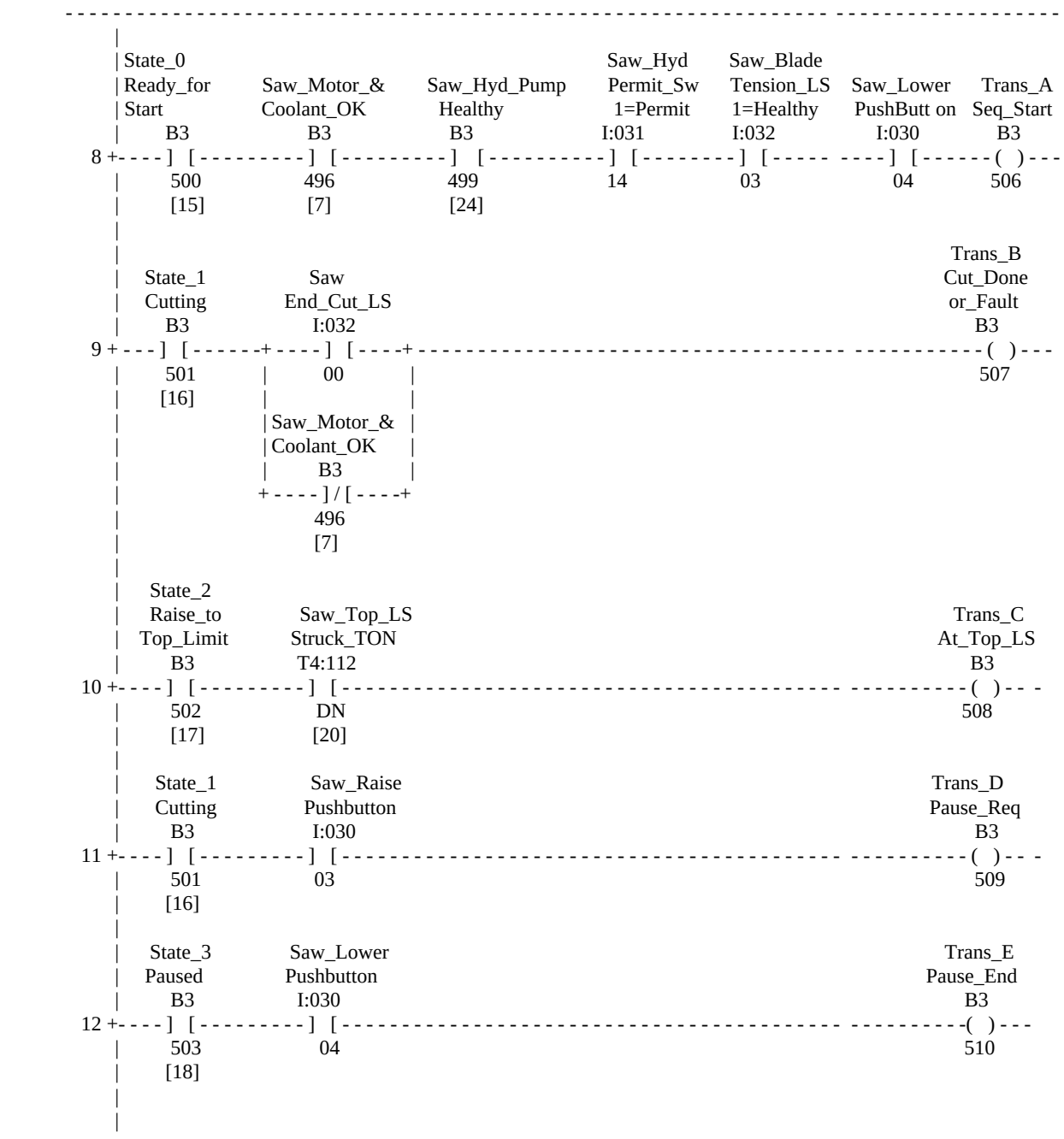


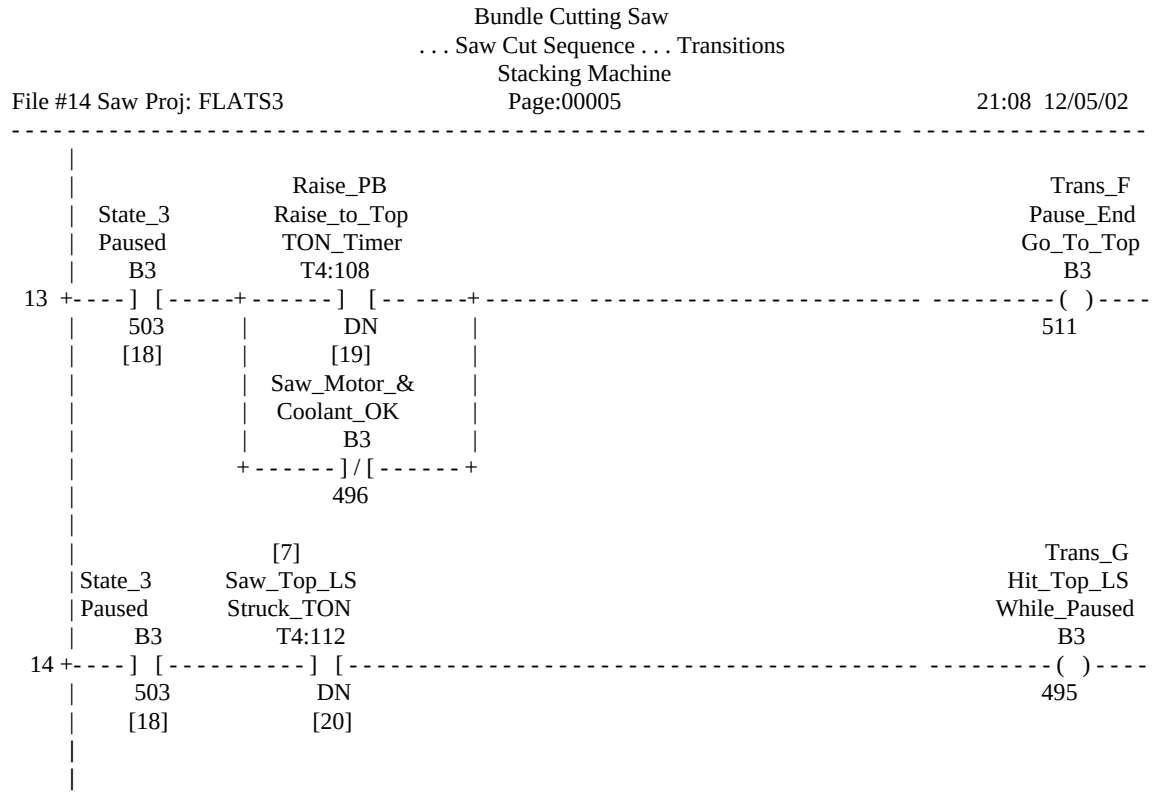
```
                                    Bundle Cutting Saw
                                     ... Coolant Pump
                                     Stacking Machine
File #14 Saw Proj: FLATS3            Page:00002                                  21:08  12/05/02
- - - -
```

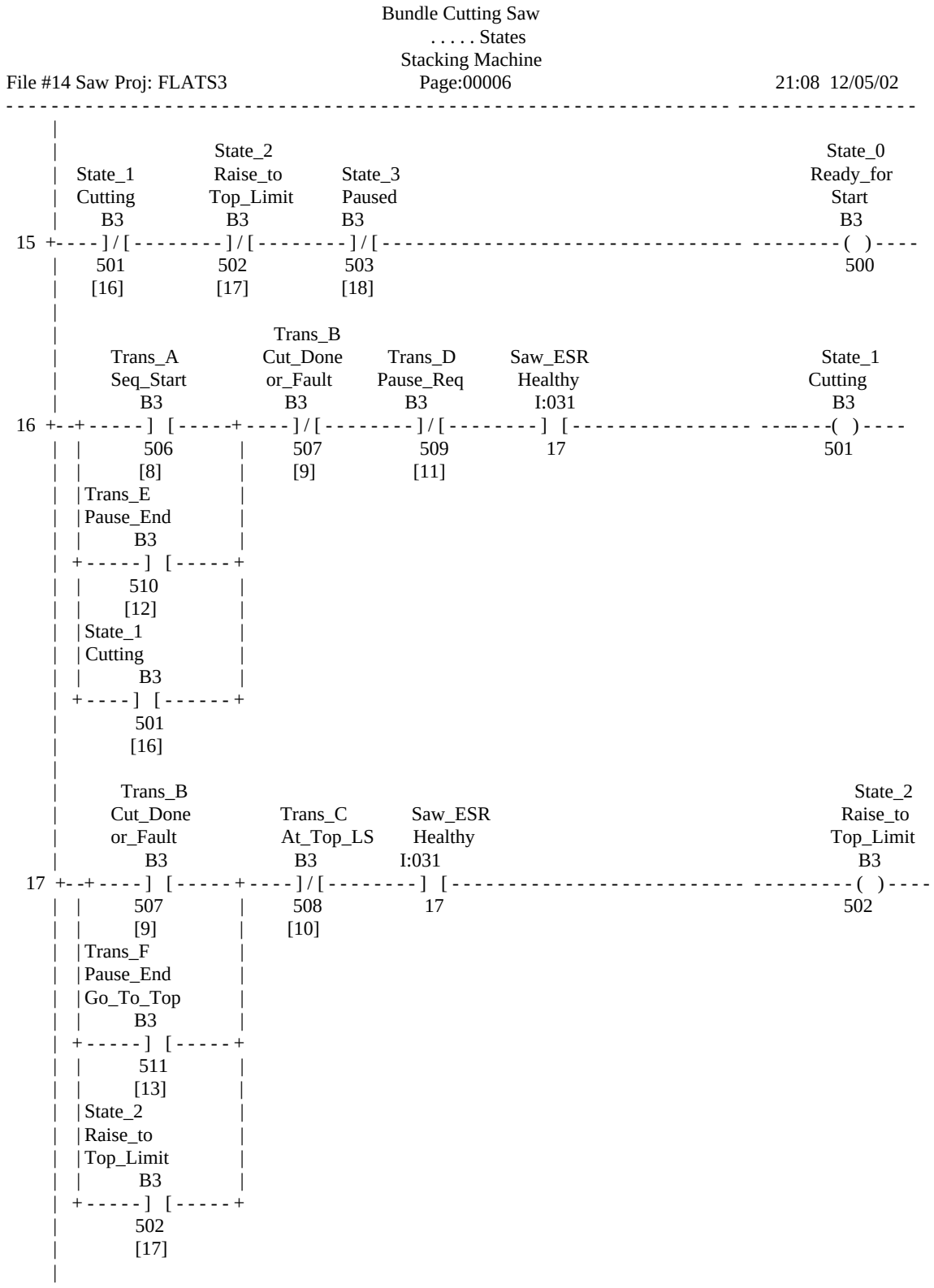
```
                                   Bundle Cutting Saw
                                      ... Coolant Pump
                                      Stacking Machine
File #14 Saw Proj: FLATS3           Page:00003                  21:08  12/05/02
- - - -
```

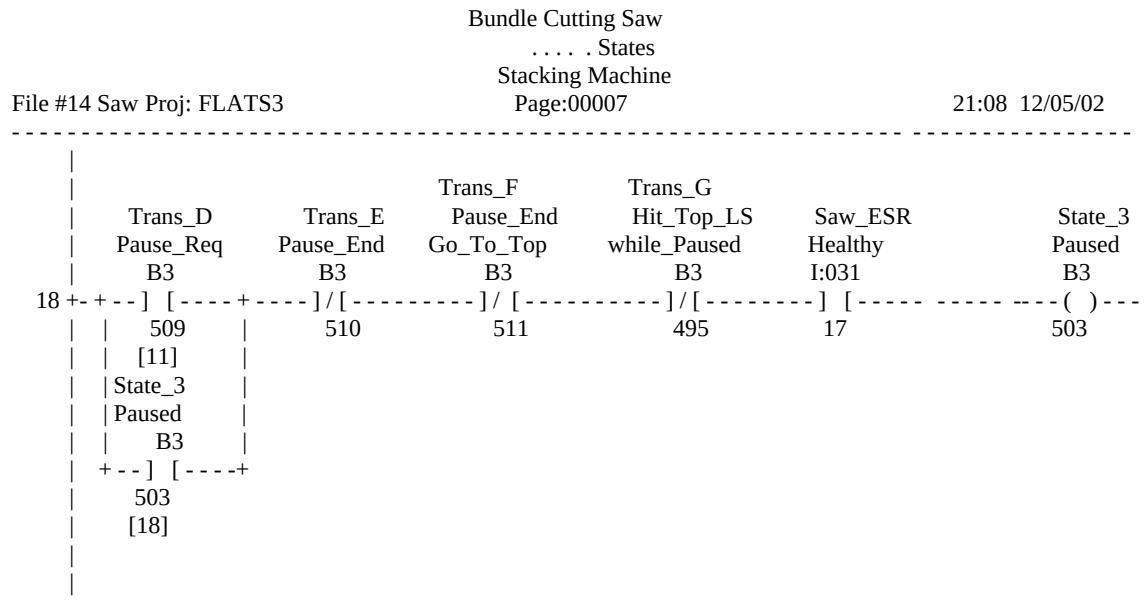
Bundle Cutting Saw
. . . Saw Cut Sequence . . . Transitions
Stacking Machine

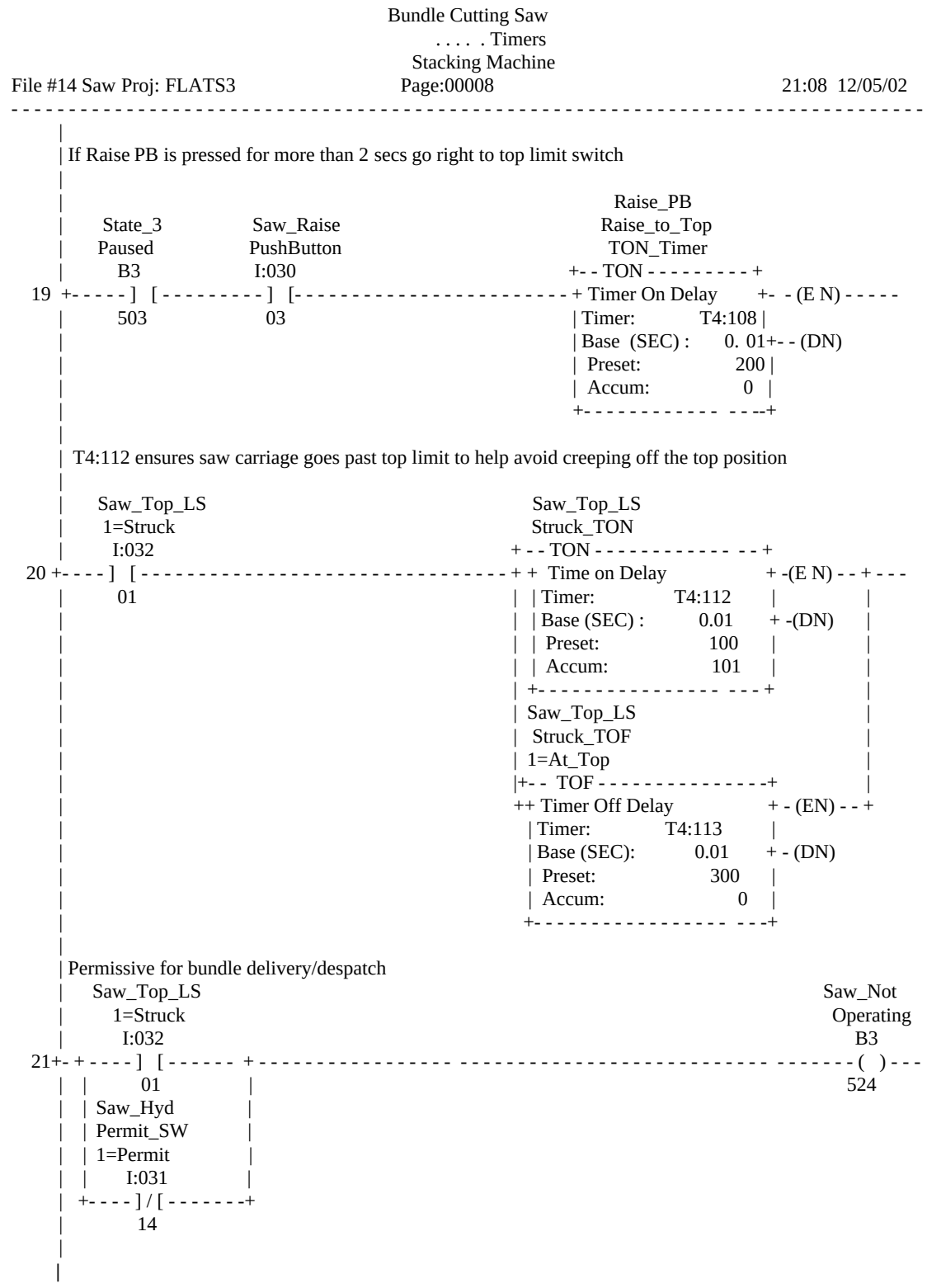
File #14 Saw Proj: FLATS3 Page:00004 21:08 12/05/02

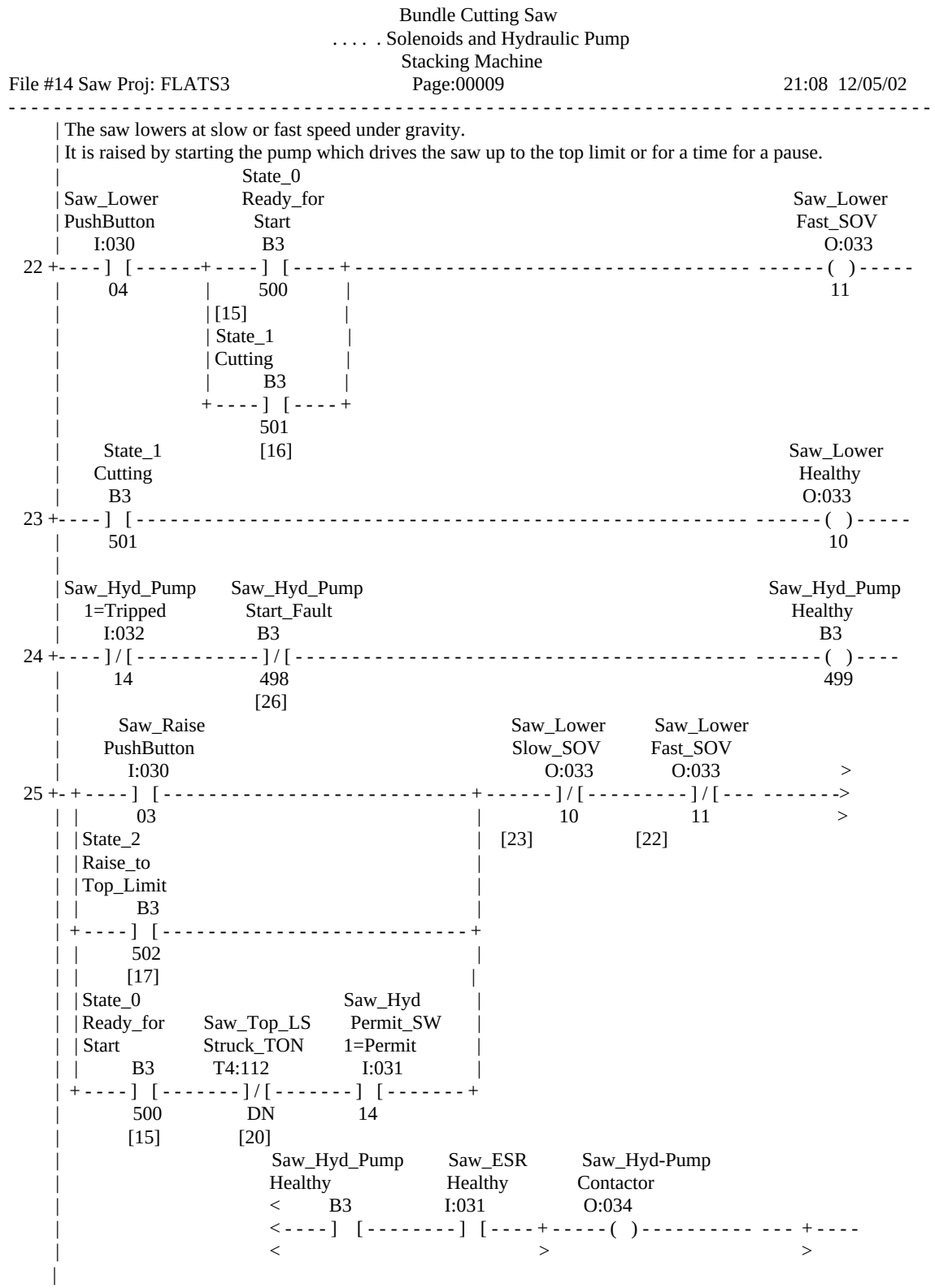


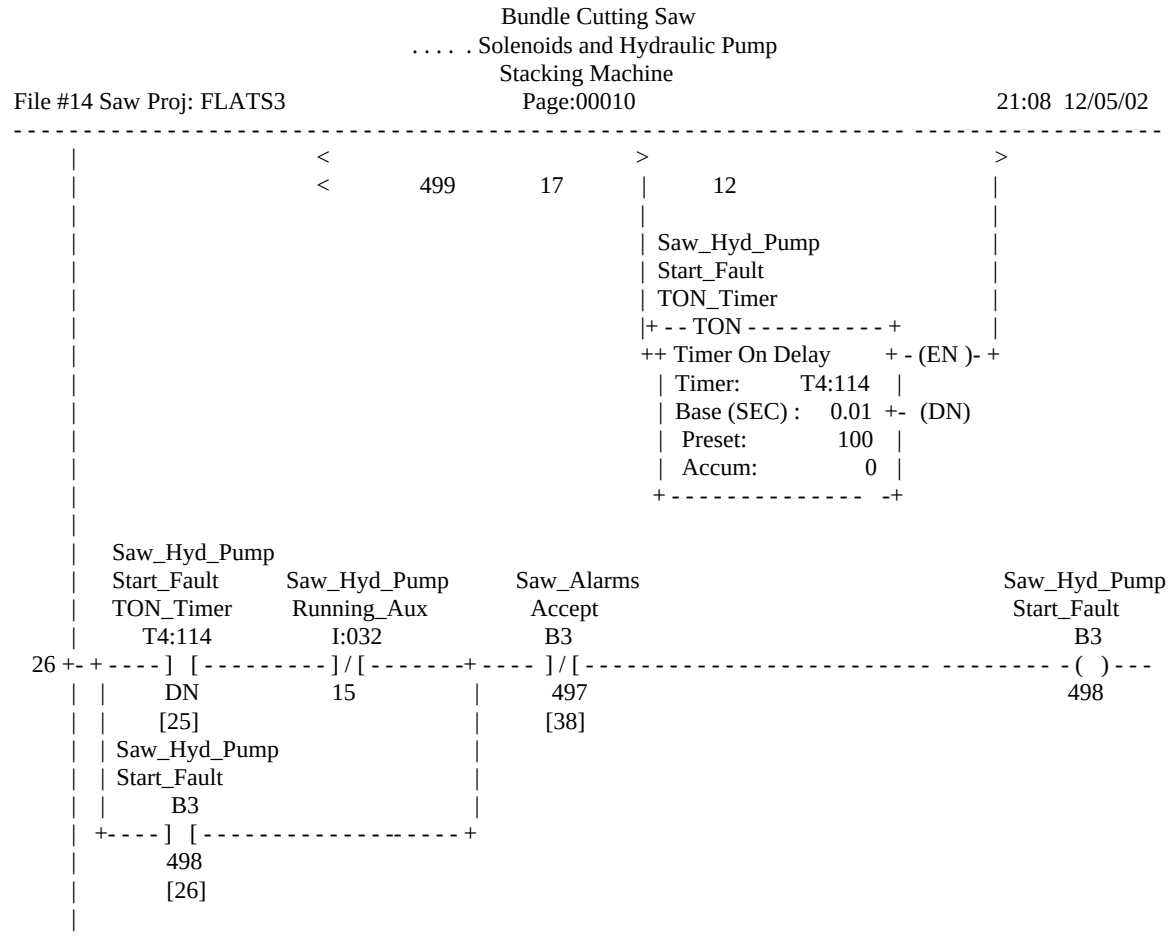

Bundle Cutting Saw
. . . Saw Cut Sequence . . . Transitions
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00005
21:08 12/05/02
State_3 Paused B3
503 [18]
Raise_PB Raise_to_Top TON_Timer T4:108
DN [19]
Saw_Motor_& Coolant_OK B3
496
Trans_F Pause_End Go_To_Top B3
511
13
State_3 Paused B3
503 [18]
[7] Saw_Top_LS Struck_TON T4:112
DN [20]
Trans_G Hit_Top_LS While_Paused B3
495
14


Bundle Cutting Saw
. . . . . States
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00006
21:08 12/05/02
State_1
Cutting
B3
501
[16]
State_2
Raise_to
Top_Limit
B3
502
[17]
State_3
Paused
B3
503
[18]
State_0
Ready_for
Start
B3
500
15
Trans_A
Seq_Start
B3
506
[8]
Trans_B
Cut_Done
or_Fault
B3
507
[9]
Trans_D
Pause_Req
B3
509
[11]
Saw_ESR
Healthy
I:031
17
State_1
Cutting
B3
501
16
Trans_E
Pause_End
B3
510
[12]
State_1
Cutting
B3
501
[16]
Trans_B
Cut_Done
or_Fault
B3
507
[9]
Trans_C
At_Top_LS
B3
508
[10]
Saw_ESR
Healthy
I:031
17
State_2
Raise_to
Top_Limit
B3
502
17
Trans_F
Pause_End
Go_To_Top
B3
511
[13]
State_2
Raise_to
Top_Limit
B3
502
[17]

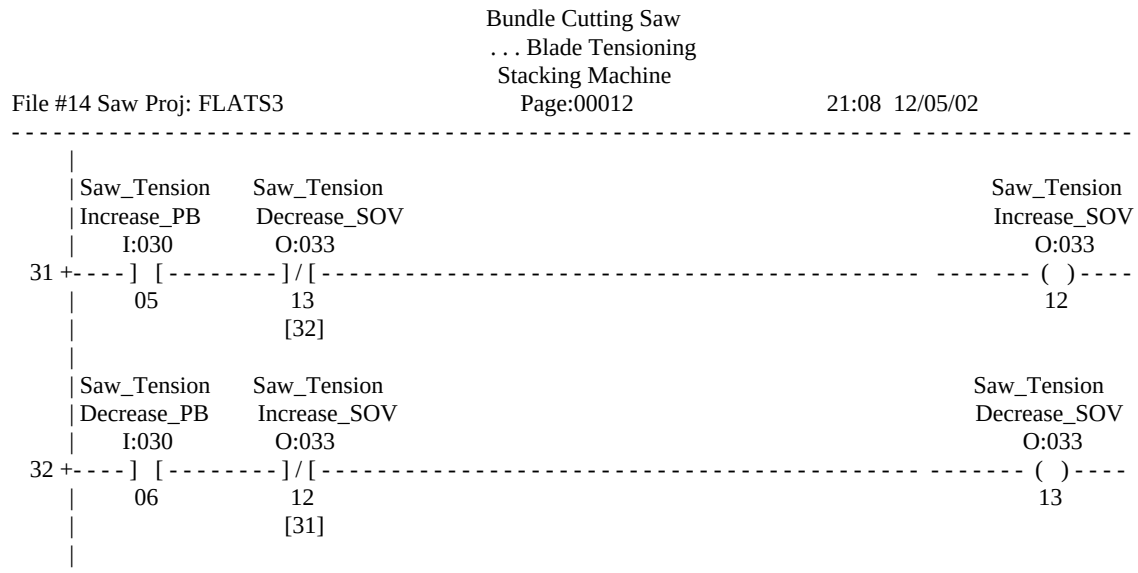

Bundle Cutting Saw
. . . . . States
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00007
21:08 12/05/02
Trans_D
Pause_Req
B3
Trans_E
Pause_End
B3
Trans_F
Pause_End
Go_To_Top
B3
Trans_G
Hit_Top_LS
while_Paused
B3
Saw_ESR
Healthy
I:031
State_3
Paused
B3
18
509
510
511
495
17
503
[11]
State_3
Paused
B3
503
[18]


Bundle Cutting Saw
. . . . . Timers
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00008
21:08 12/05/02
If Raise PB is pressed for more than 2 secs go right to top limit switch
State_3
Paused
B3
503
Saw_Raise
PushButton
I:030
03
Raise_PB
Raise_to_Top
TON_Timer
TON
Timer On Delay
Timer: T4:108
Base (SEC) : 0. 01
Preset: 200
Accum: 0
(E N)
(DN)
19
T4:112 ensures saw carriage goes past top limit to help avoid creeping off the top position
Saw_Top_LS
1=Struck
I:032
01
Saw_Top_LS
Struck_TON
TON
Time on Delay
Timer: T4:112
Base (SEC) : 0.01
Preset: 100
Accum: 101
(E N)
(DN)
Saw_Top_LS
Struck_TOF
1=At_Top
TOF
Timer Off Delay
Timer: T4:113
Base (SEC): 0.01
Preset: 300
Accum: 0
(EN)
(DN)
20
Permissive for bundle delivery/despatch
Saw_Top_LS
1=Struck
I:032
01
Saw_Hyd
Permit_SW
1=Permit
I:031
14
Saw_Not
Operating
B3
524
21

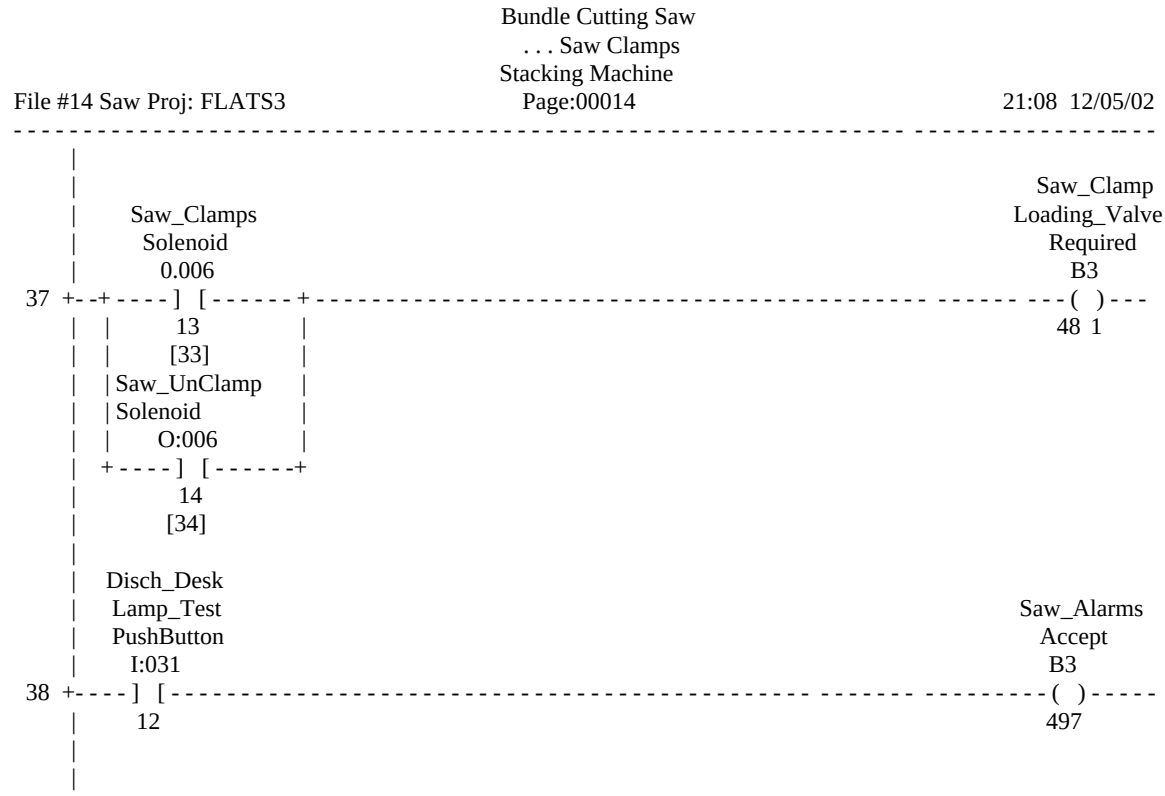

Bundle Cutting Saw
. . . . . Solenoids and Hydraulic Pump
Stacking Machine
File #14 Saw Proj: FLATS3 Page:00009 21:08 12/05/02
The saw lowers at slow or fast speed under gravity.
It is raised by starting the pump which drives the saw up to the top limit or for a time for a pause.
Saw_Lower PushButton I:030 04
State_0 Ready_for Start B3 500
Saw_Lower Fast_SOV O:033 11
[15]
State_1 Cutting B3
501
State_1 Cutting B3 501
[16]
Saw_Lower Healthy O:033 10
Saw_Hyd_Pump 1=Tripped I:032 14
Saw_Hyd_Pump Start_Fault B3 498
Saw_Hyd_Pump Healthy B3 499
[26]
Saw_Raise PushButton I:030 03
Saw_Lower Slow_SOV O:033 10
Saw_Lower Fast_SOV O:033 11
[23]
[22]
State_2 Raise_to Top_Limit B3 502
[17]
State_0 Ready_for Start B3 500
[15]
Saw_Top_LS Struck_TON T4:112 DN
[20]
Saw_Hyd Permit_SW 1=Permit I:031 14
Saw_Hyd_Pump Healthy B3
Saw_ESR Healthy I:031
Saw_Hyd-Pump Contactor O:034

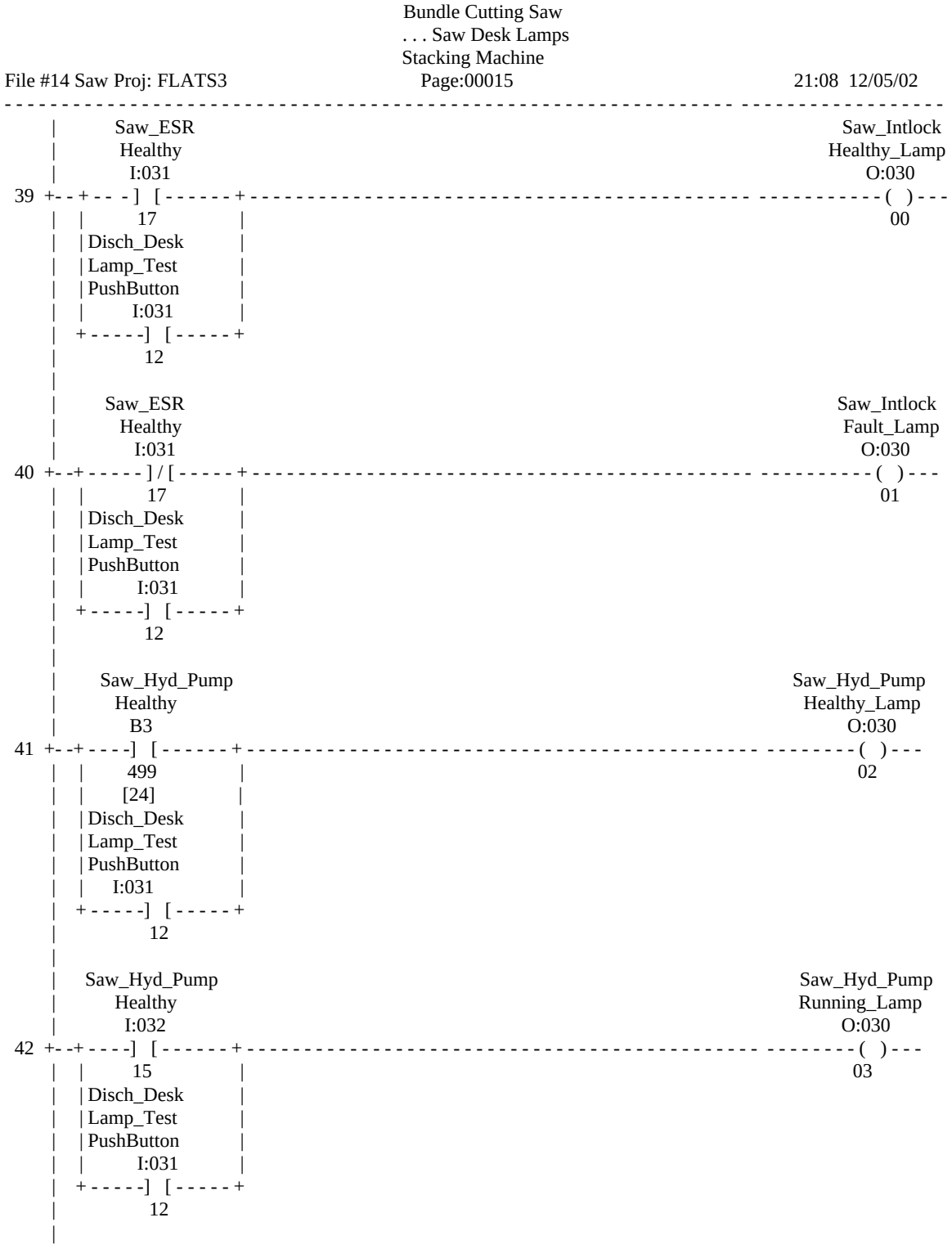

Bundle Cutting Saw
. . . . . Solenoids and Hydraulic Pump
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00010
21:08 12/05/02
499
17
12
Saw_Hyd_Pump
Start_Fault
TON_Timer
TON
Timer On Delay
EN
Timer: T4:114
Base (SEC) : 0.01
DN
Preset: 100
Accum: 0
Saw_Hyd_Pump
Start_Fault
TON_Timer
T4:114
Saw_Hyd_Pump
Running_Aux
I:032
Saw_Alarms
Accept
B3
Saw_Hyd_Pump
Start_Fault
B3
26
DN
15
497
498
[25]
[38]
Saw_Hyd_Pump
Start_Fault
B3
498
[26]

```
                                        Bundle Cutting Saw
                                     ... Blade Tensioning
                                        Stacking Machine
File #14 Saw Proj: Flats3                Page:00011                                  21:08  12/05/02
- - - -
```


Bundle Cutting Saw
. . . Blade Tensioning
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00012
21:08 12/05/02
Saw_Tension Increase_PB I:030 05
Saw_Tension Decrease_SOV O:033 13 [32]
Saw_Tension Increase_SOV O:033 12
Saw_Tension Decrease_PB I:030 06
Saw_Tension Increase_SOV O:033 12 [31]
Saw_Tension Decrease_SOV O:033 13

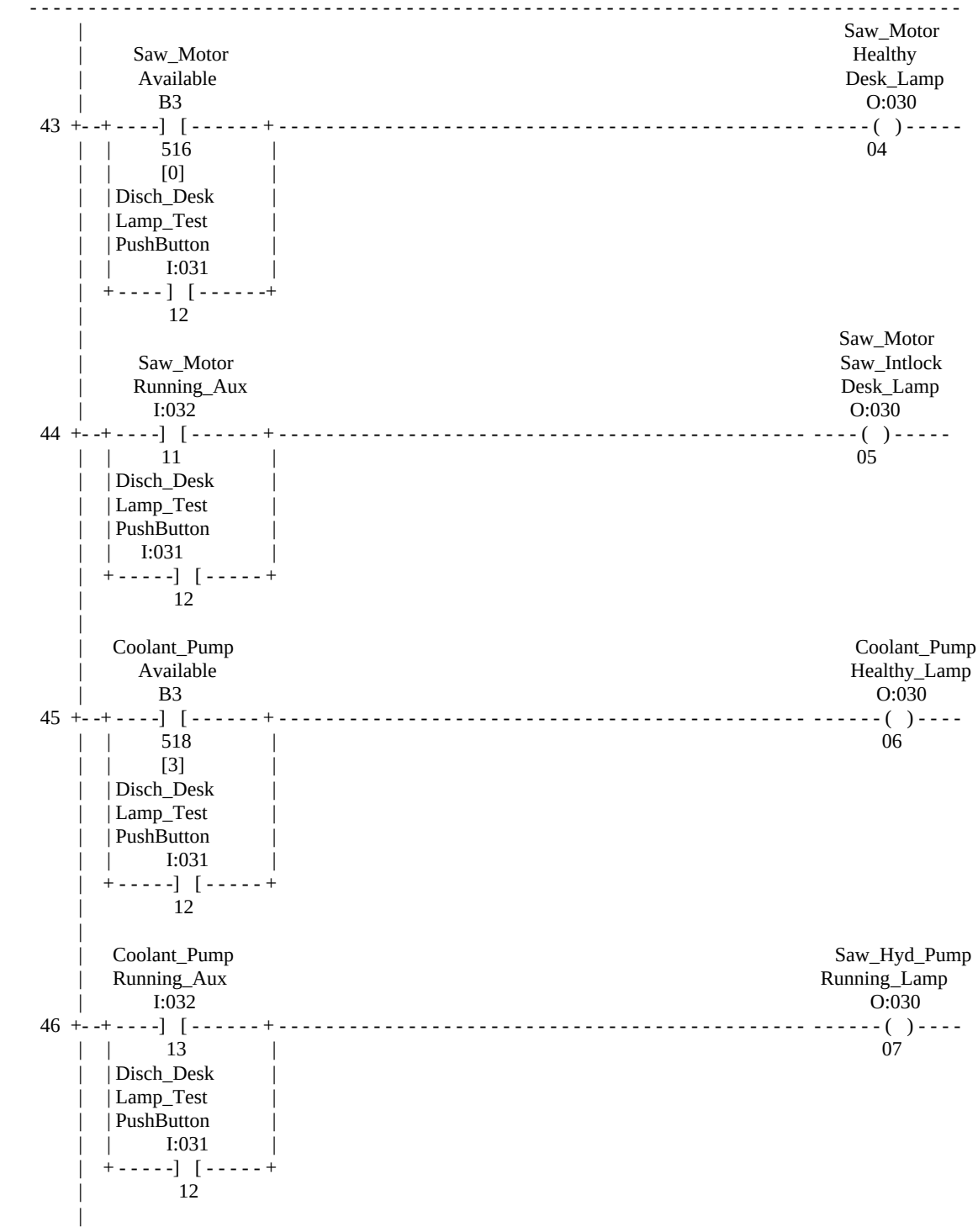
```
                                        Bundle Cutting Saw
                                        ... Saw Clamps
                                        Stacking Machine
File #14 Saw Proj: FLATS3                    Page:00013                              21:08  12/05/02
- - - -
```

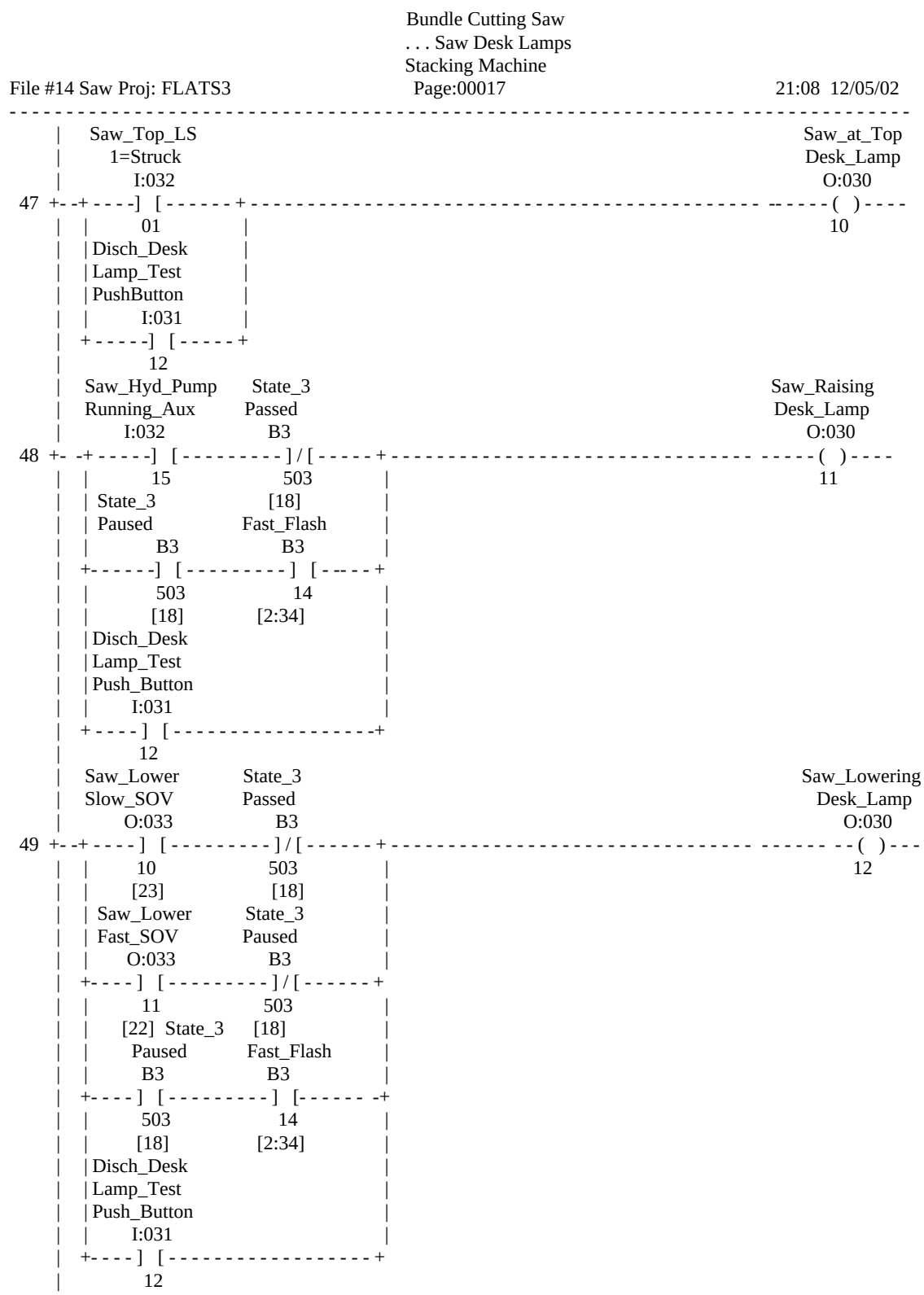

Bundle Cutting Saw
. . . Saw Clamps
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00014
21:08 12/05/02
Saw_Clamps
Solenoid
0.006
13
[33]
Saw_UnClamp
Solenoid
O:006
14
[34]
Saw_Clamp
Loading_Valve
Required
B3
48 1
Disch_Desk
Lamp_Test
PushButton
I:031
12
Saw_Alarms
Accept
B3
497

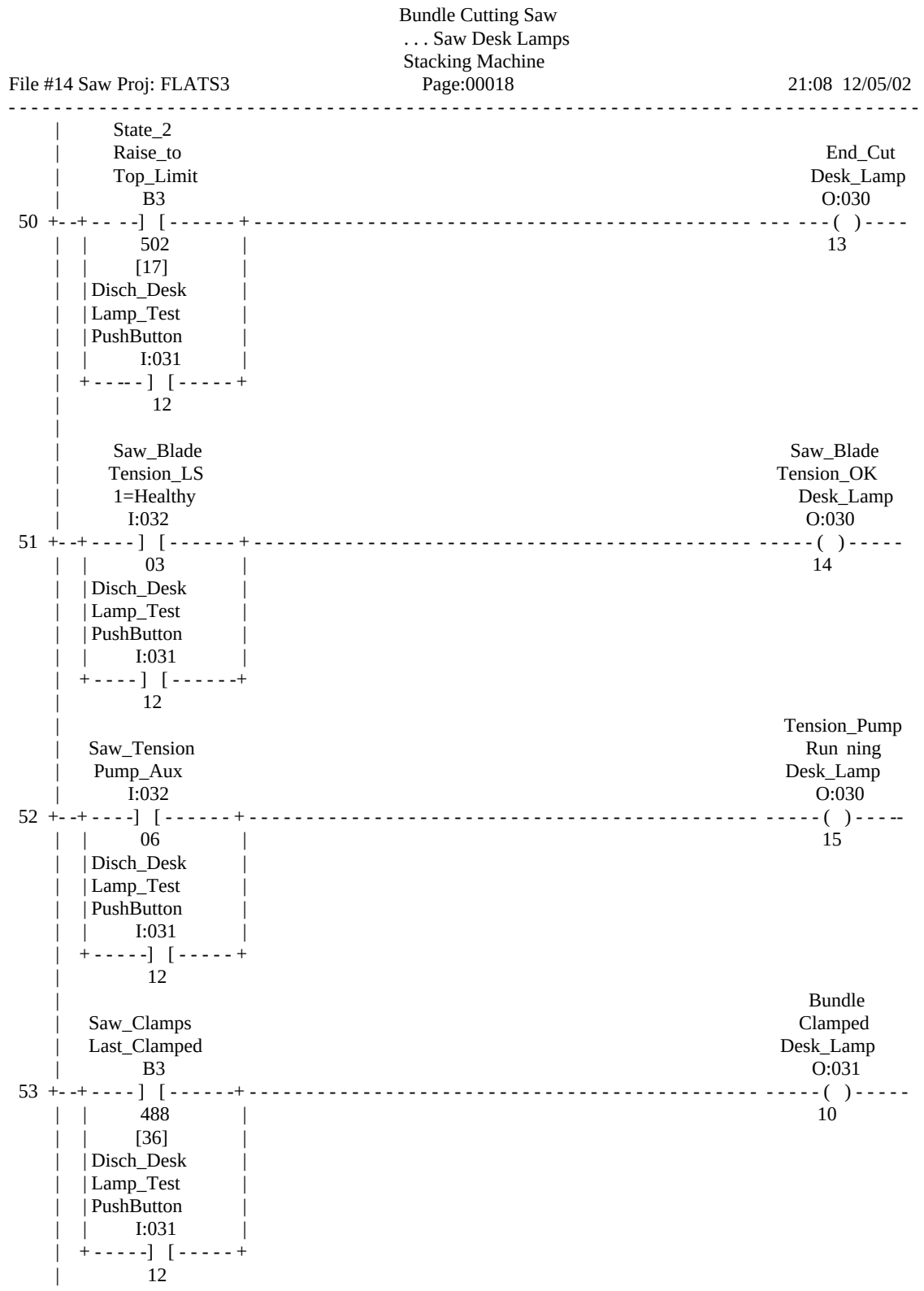

Bundle Cutting Saw
. . . Saw Desk Lamps
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00015
21:08 12/05/02
Saw_ESR
Healthy
I:031
Saw_Intlock
Healthy_Lamp
O:030
39
17
Disch_Desk
Lamp_Test
PushButton
I:031
12
00
Saw_ESR
Healthy
I:031
Saw_Intlock
Fault_Lamp
O:030
40
17
Disch_Desk
Lamp_Test
PushButton
I:031
12
01
Saw_Hyd_Pump
Healthy
B3
Saw_Hyd_Pump
Healthy_Lamp
O:030
41
499
[24]
Disch_Desk
Lamp_Test
PushButton
I:031
12
02
Saw_Hyd_Pump
Healthy
I:032
Saw_Hyd_Pump
Running_Lamp
O:030
42
15
Disch_Desk
Lamp_Test
PushButton
I:031
12
03

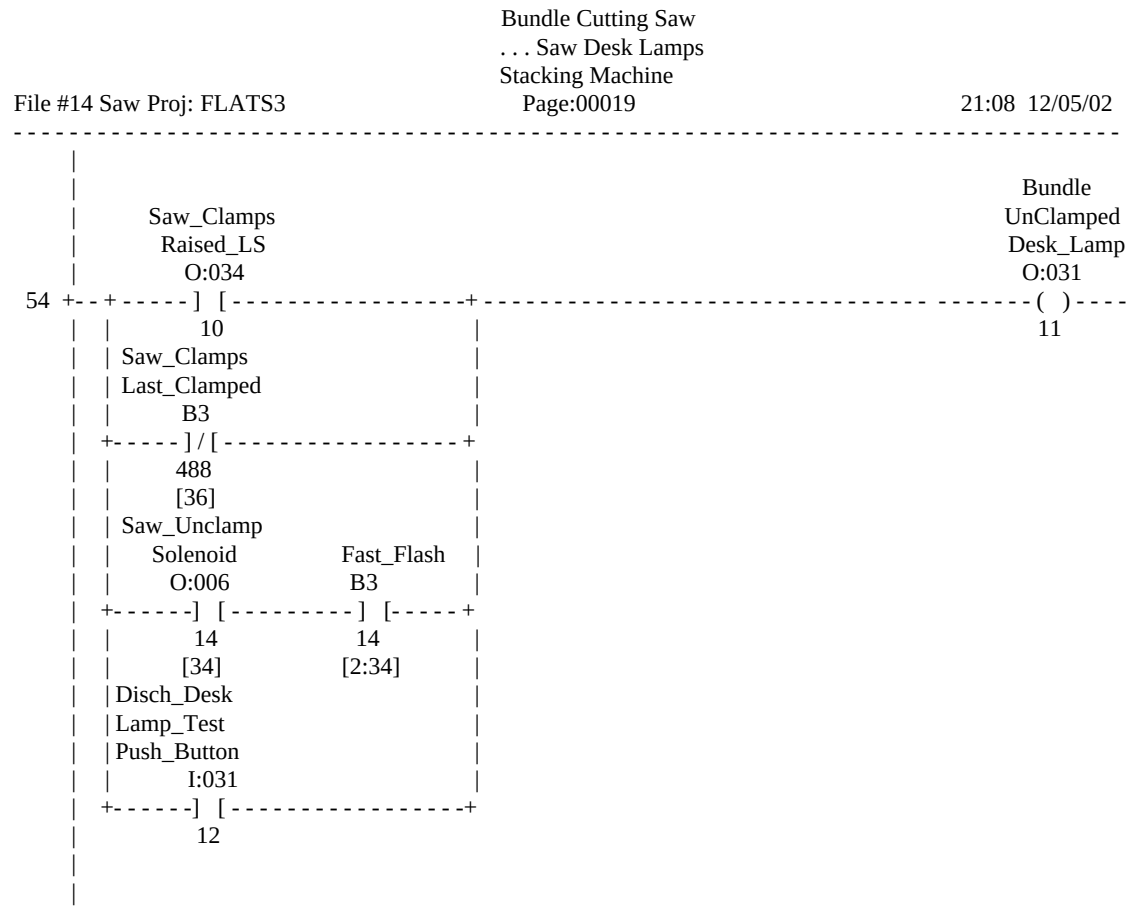
Bundle Cutting Saw
. . . Saw Desk Lamps
Stacking Machine

File #14 Saw Proj: FLATS3 Page:00016 21:08 12/05/02

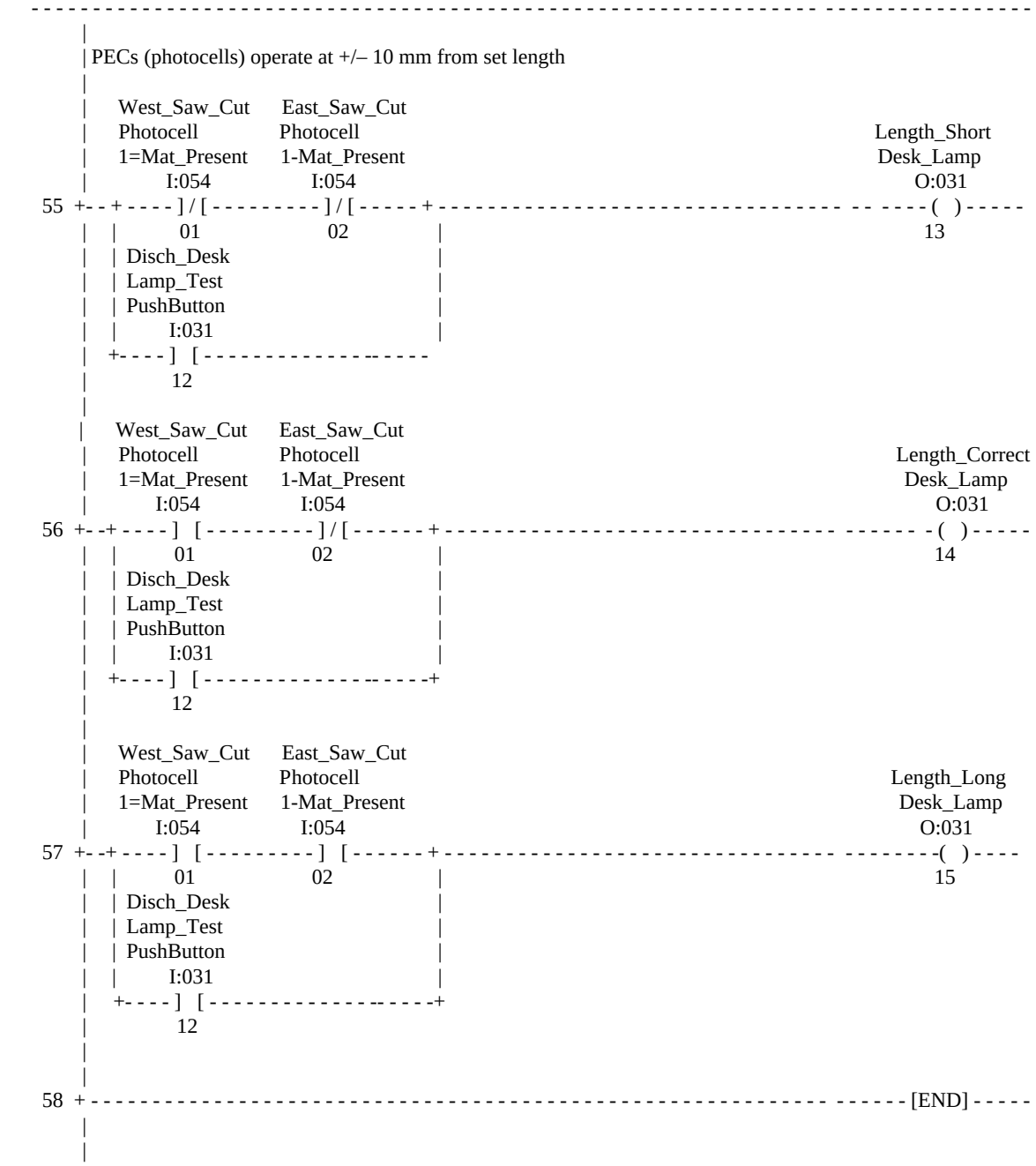



Bundle Cutting Saw
. . . Saw Desk Lamps
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00017
21:08 12/05/02
Saw_Top_LS
1=Struck
I:032
Saw_at_Top
Desk_Lamp
O:030
47
01
Disch_Desk
Lamp_Test
PushButton
I:031
10
12
Saw_Hyd_Pump
Running_Aux
I:032
State_3
Passed
B3
Saw_Raising
Desk_Lamp
O:030
48
15
503
State_3
Paused
B3
[18]
Fast_Flash
B3
11
503
14
[18]
[2:34]
Disch_Desk
Lamp_Test
Push_Button
I:031
12
Saw_Lower
Slow_SOV
O:033
State_3
Passed
B3
Saw_Lowering
Desk_Lamp
O:030
49
10
503
[23]
[18]
Saw_Lower
Fast_SOV
O:033
State_3
Paused
B3
12
11
503
[22]
State_3
Paused
B3
[18]
Fast_Flash
B3
503
14
[18]
[2:34]
Disch_Desk
Lamp_Test
Push_Button
I:031
12


Bundle Cutting Saw
. . . Saw Desk Lamps
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00018
21:08 12/05/02
State_2
Raise_to
Top_Limit
B3
502
[17]
Disch_Desk
Lamp_Test
PushButton
I:031
12
End_Cut
Desk_Lamp
O:030
13
Saw_Blade
Tension_LS
1=Healthy
I:032
03
Disch_Desk
Lamp_Test
PushButton
I:031
12
Saw_Blade
Tension_OK
Desk_Lamp
O:030
14
Saw_Tension
Pump_Aux
I:032
06
Disch_Desk
Lamp_Test
PushButton
I:031
12
Tension_Pump
Run ning
Desk_Lamp
O:030
15
Saw_Clamps
Last_Clamped
B3
488
[36]
Disch_Desk
Lamp_Test
PushButton
I:031
12
Bundle
Clamped
Desk_Lamp
O:031
10


Bundle Cutting Saw
. . . Saw Desk Lamps
Stacking Machine
File #14 Saw Proj: FLATS3
Page:00019
21:08 12/05/02
Saw_Clamps
Raised_LS
O:034
Bundle
UnClamped
Desk_Lamp
O:031
54
10
11
Saw_Clamps
Last_Clamped
B3
488
[36]
Saw_Unclamp
Solenoid
O:006
Fast_Flash
B3
14
[34]
14
[2:34]
Disch_Desk
Lamp_Test
Push_Button
I:031
12

Bundle Cutting Saw
. . . Saw Cutting Length Lamps (from PECs)
Stacking Machine
File #14 Saw Proj: FLATS3 Page:00020 21:08 12/05/02



Problems

Questions 1 to 6 have four answer options: A, B, C or D. Choose the correct answer from the answer options.

1 Decide whether each of these statements is True (T) or False (F).

The term forcing when applied to a PLC input/output means using a program to:
(i) Turn on or off inputs/outputs.
(ii) Check that all inputs/outputs give correct responses when selected.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

2 Decide whether each of these statements is True (T) or False (F).

The term watchdog when applied to a PLC means a checking mechanism that:
(i) Excessive currents are not occurring.
(ii) Functions are carried out within prescribed time limits.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

3 Decide whether each of these statements is True (T) or False (F).

When a PLC is in monitor/test/debug mode it:
(i) Enables the operation of a program to be simulated.
(ii) Carries out a fault check.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

4 When a PLC is in monitor/test/debug mode and the symbol shown in Figure 13.20 occurs, it means that an input is:

Figure 13.20 *Problem 4*

A Defective.
B Correctly operating.
C On.
D Off.

5 Decide whether each of these statements is True (T) or False (F).

Failure of an input sensor or its wiring, rather than failure of an LED or in the PLC input channel, will show as:
(i) The input LED not coming on.
(ii) Forcing of that input making the input LED come on.

A (i) T (ii) T
B (i) T (ii) F
C (i) F (ii) T
D (i) F (ii) F

6 A single output device fails to turn on when the output LED is on. The voltage at the output is tested and found normal but the voltage at the device is found to be absent. The fault is:

A Faulty wiring.
B A faulty output device.
C A fault in the PLC.
D A fault in the program.

7 Explain how, using forcing, the failure of an input sensor or its wiring can be detected.

8 Suggest possible causes of a complete stoppage of the control operation and the PLC with the power on lamp off.

9 Suggest possible causes of an output LED being on but the output device failing to turn on.

10 Devise a timing watchdog program to be used to switch off a machine if faults occur in any of the systems controlling its actions.

11 Design the program for a pneumatic system for control by a PLC to give the cylinder sequence A+, B+, B–, A– and which will give a LED display indicating, in the presence of a fault such as a sticking cylinder, at which point in the cycle the fault occurred. Explain the action of all elements in the system.

14 Programs

This chapter extends the examples given in previous chapters to show programs developed to complete specific tasks. These include tasks which involve temperature control and a number involving pneumatic valves.

14.1 Temperature control

Consider the task of using a PLC as an on-off controller for a heater in the control of temperature in some enclosure. The heater is to be switched on when the temperature falls below the required temperature and switched off when the temperature is at or above the required temperature. The basic algorithm might be considered to be:

```
IF temperature below set value
THEN
    DO switch on heater
ELSE
    DO switch off heater
ENDIF
```

The sensor used for the temperature might be a thermocouple, a thermistor or an integrated chip (see Section 2.1.5). When connected in an appropriate circuit, the sensor will give a suitable voltage signal related to the temperature. This voltage can be compared, using an operational amplifier, with the voltage set for the required temperature so that a high output signal is given when the temperature is above the required temperature and a low output signal when it is below. Thus when the temperature falls from above the required temperature to below it, the signal switches from a high to a low value. This transition can be used as the input to a PLC. The PLC can then be programmed to give an output when there is a low input and this output used to switch on the heater.

Figure 14.1 shows the arrangement that might be used and a Mitsubishi ladder program. The input from the operational amplifier has been connected to the input port with the address X400. This input has contacts which are normally closed. When the input goes high, the contacts open. The output is taken from the output port with the address Y430. Thus there is an output when the input is low and no output when the input is high.

Figure 14.2 shows the key sequence that would be used with a graphic programmer to enter the part of the program given in Figure 14.1.

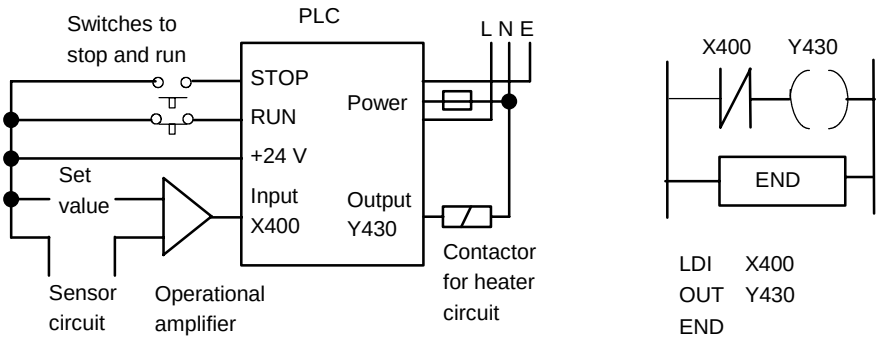


Figure 14.1 *Temperature control*

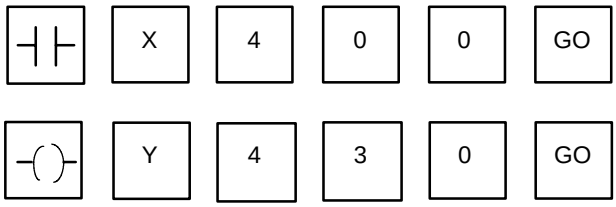


Figure 14.2 *Entering program graphically*

In Siemens format the program given in Figure 14.1 could be as shown in Figure 14.3 and in Allen-Bradley form as shown in Figure 14.4. To illustrate how such a program might be entered using a computer, Figure 14.5 also shows the function keys that would be used to enter that part of the program with the Allen-Bradley software loaded. At each instant in the program a screen displays prompts and lists the function keys and their significance.

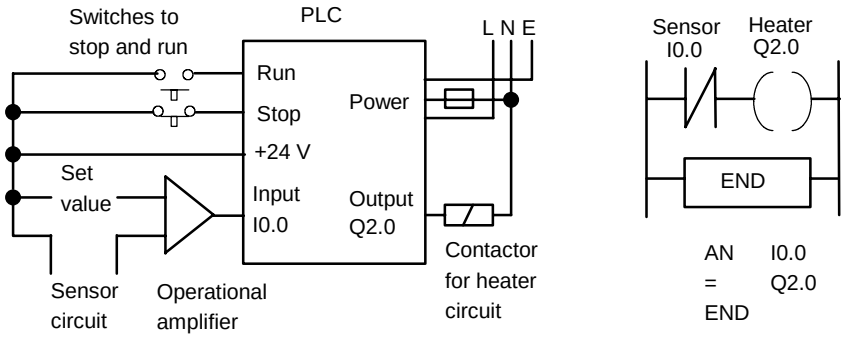
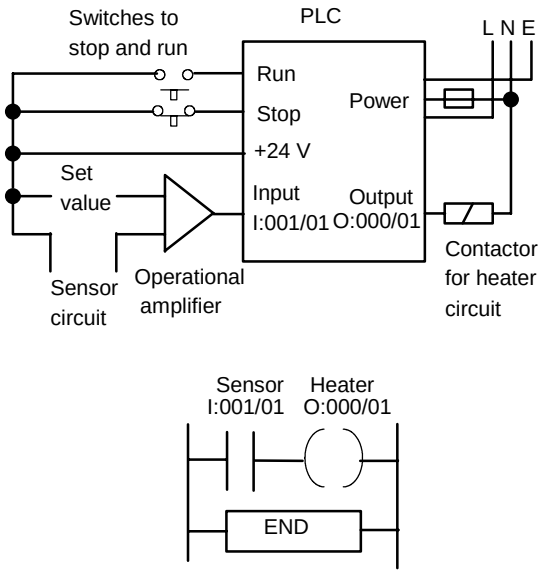


Figure 14.3 *Temperature control*



Offline programming selected F1
Mode selected: Create file F6
File name entered: TEMP, enter key pressed
Select the processor: F2 pressed until relevant one on screen.
Create file F1
Monitor file F8. This shows the file as just consisting of the END rung.
Edit F10. This is to enable the file to be modified.
Append rung and append instruction F3 pressed twice.
-||- F1 key pressed.
I;001/01 entered, enter key pressed.
-( )- F3 key pressed.
O;000/01 entered, enter key pressed.
To accept the program Esc is pressed.

Figure 14.4 *Allen-Bradley program*

Consider a more complex temperature control task involving a domestic central heating system (Figure 14.5). The central heating boiler is to be thermostatically controlled and supply hot water to the radiator system in the house and also to the hot water tank to provide hot water from the taps in the house. Pump motors have to be switched on to direct the hot water from the boiler to either, or both the radiator and hot water systems according to whether the temperature sensors for the room temperature and the hot water tank indicate that the radiators or tank need heating. The entire system is to be controlled by a clock so that it only operates for certain hours of the day. Figure 14.6(a) shows how a Mitsubishi PLC, and Figure 14.6(b) a Siemens PLC, might be used.

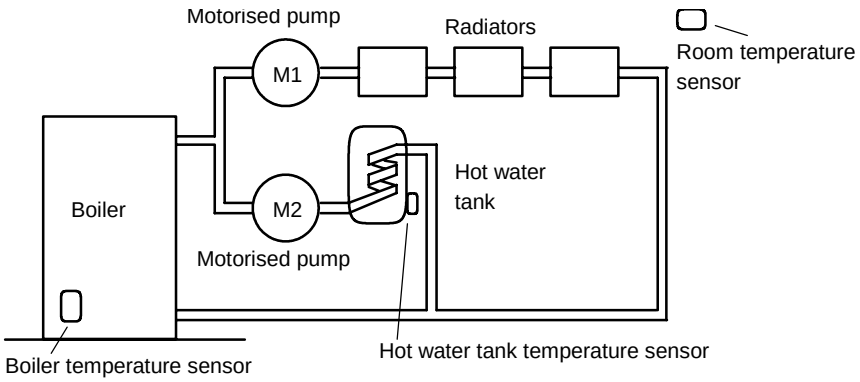
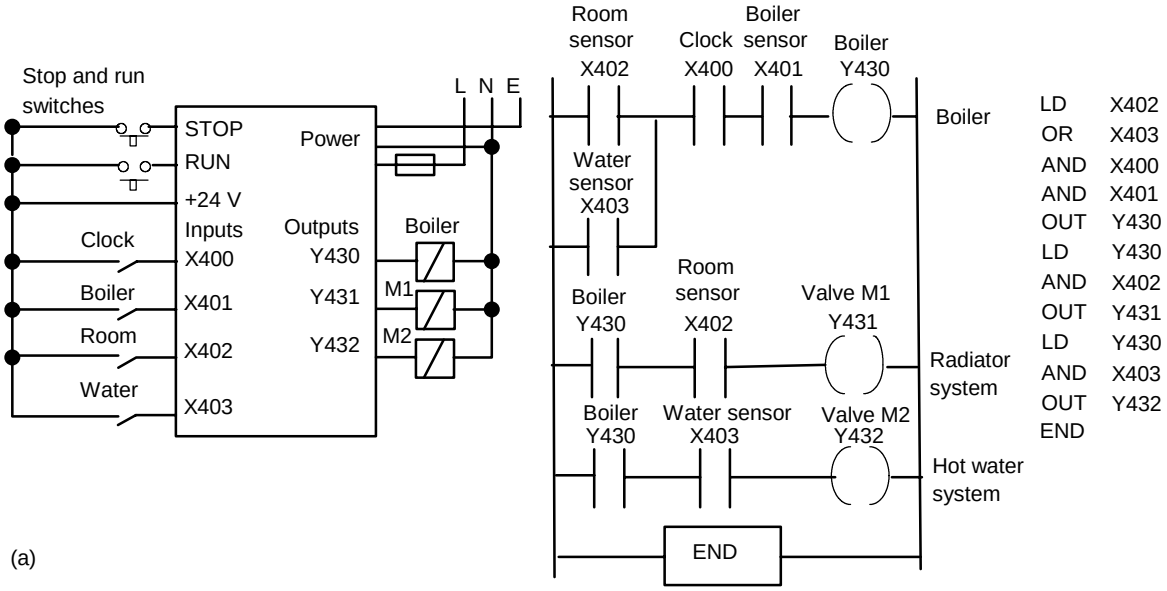
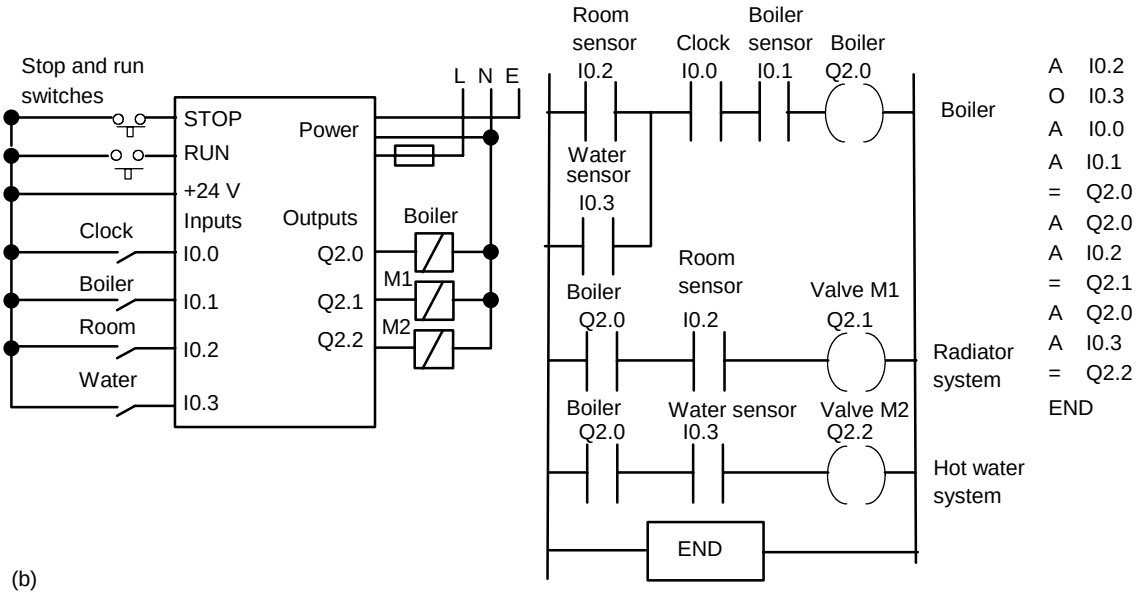


Figure 14.5 *Central heating system*

The boiler, output Y430/Q2.0, is switched on if X400/I0.0 and X401/I0.1 and either X402/I0.2 or X403/I0.3 are switched on. This means if the clock is switched on, the boiler temperature sensor gives an on input, and either the room temperature sensor or the water temperature sensor give on inputs. The motorised valve M1, output Y431/Q2.1, is switched on if the boiler, Y430/Q2.0, is on and if the room temperature sensor X402/I0.2 gives an on input. The motorised valve M2, output Y432/Q2.2, is switched on if the boiler, Y430/Q2.0, is on and if the water temperature sensor gives an on input.



(a)



(b)

Figure 14.6 *Central heating system: (a) Mitsubishi, (b) Siemens*

In the above discussion a simple on/off form of temperature control has been used, a comparator op-amp giving a 1 output when the temperature is above the set temperature and 0 when it is below. The output to the heating system is then just on or off. A more elaborate system is to use proportional control with the output to the heating system being a signal proportional to the difference in temperature between that occurring and the set value. The program might then carry out the following tasks:

1 Read the input actual temperature after conversion from analogue to digital by an ADC.
2 Input the set point temperature.
3 Subtract the actual temperature from the set point temperature.
4 Multiply the result by the proportional constant.
5 Use the result to control the value of the output to the heater.

14.2 Valve sequencing

Consider tasks involving directional control valves (see Section 2.2.2 for an introductory discussion). Directional control valves (see Section 2.2.2) are specified in terms of the number of ports and number of control positions they have. Figure 14.7(a) shows a 4/2 valve; when the push button is pressed, port A is connected to T, the symbol of a vent to the atmosphere or return of hydraulic fluid to the sump, and port P is connected to B, the symbol for the pressure supply. Hence, pushing the button causes air or hydraulic fluid to flow from P to B and vented to the atmosphere or returned to the sump via the connection of A to T. When the push button is released, the spring pushes the connections to the state indicated in the box attached to the spring. Thus port B is now connected to T and port P to A. The air or hydraulic fluid now flows from P to A and is vented to the atmosphere or returned to the sump via B. Alternatively we might have the valve controlled by two solenoids (Figure 14.8).

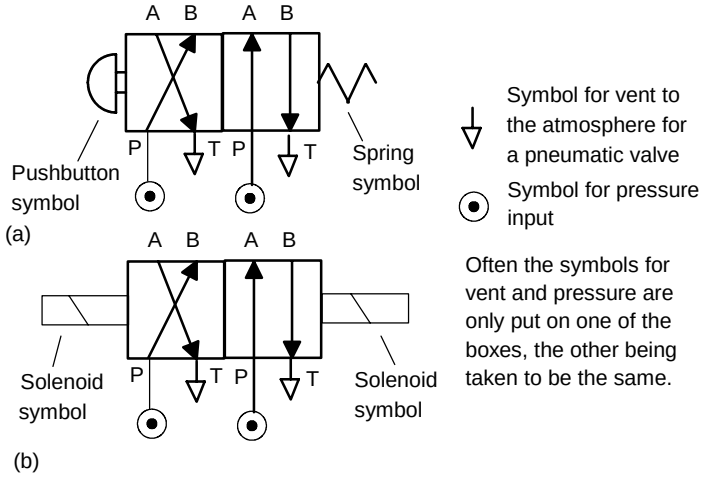

A B A B
P T P T
Pushbutton symbol
Spring symbol
(a)
A B A B
P T P T
Solenoid symbol
Solenoid symbol
(b)
Symbol for vent to the atmosphere for a pneumatic valve
Symbol for pressure input
Often the symbols for vent and pressure are only put on one of the boxes, the other being taken to be the same.

Figure 14.7 *4/2 valve*

14.2.1 Cyclic movement

Consider the task of obtaining cyclic movement of a piston in a cylinder. This might be to periodically push workpieces into position in a machine

tool with another similar, but out of phase, arrangement being used to remove completed workpieces. Figure 14.8 shows the valve and piston arrangement that might be used, a possible ladder program and chart indicating the timing of each output.

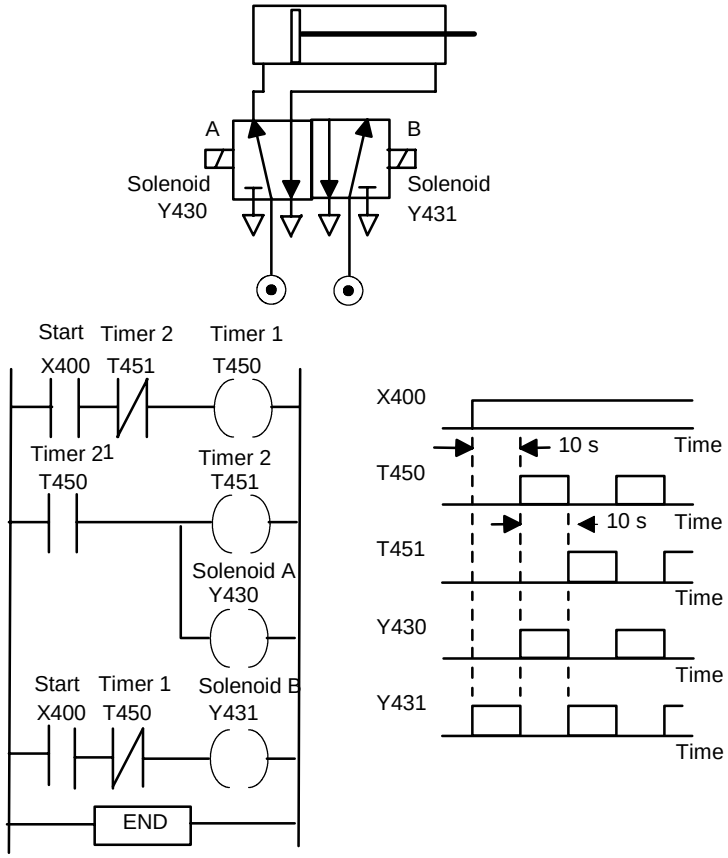


Figure 14.8 *Cyclic movement of a piston*

Consider both timers set for 10 s. When the start contacts X400 are closed, timer T450 starts. Also there is an output from Y431. The output Y431 is one of the solenoids used to actuate the valve. When it is energised it causes the pressure supply P to be applied to the right-hand end of the cylinder and the left-hand side to be connected to the vent to the atmosphere. The piston thus moves to the left. After 10 s, the normally open T450 contacts close and the normally closed T450 contacts open. This stops the output Y431, starts the timer T451 and energises the output Y430. As a result, the pressure supply P is applied to the left-hand side of the piston and the right-hand side connected to the vent to the atmosphere. The piston now moves to the right. After 10 s, the T451 normally closed contacts are opened. This causes the normally closed contacts of T450 to close and so Y431 is energised. Thus the sequence repeats itself.

14.2.2 Sequencing

Consider another task involving three pistons A, B and C that have to be actuated in the sequence: A to the right, A to the left, B to the right, B to the left, C to the right, C to the left (such a sequence is often written A+, A–, B+, B–, C+, C–). Figure 14.9 illustrates the valves that might be used and Figures 14.10 and Figure 14.11 ladder programs that might be used involving timers. An alternative would involve the use of a shift register.

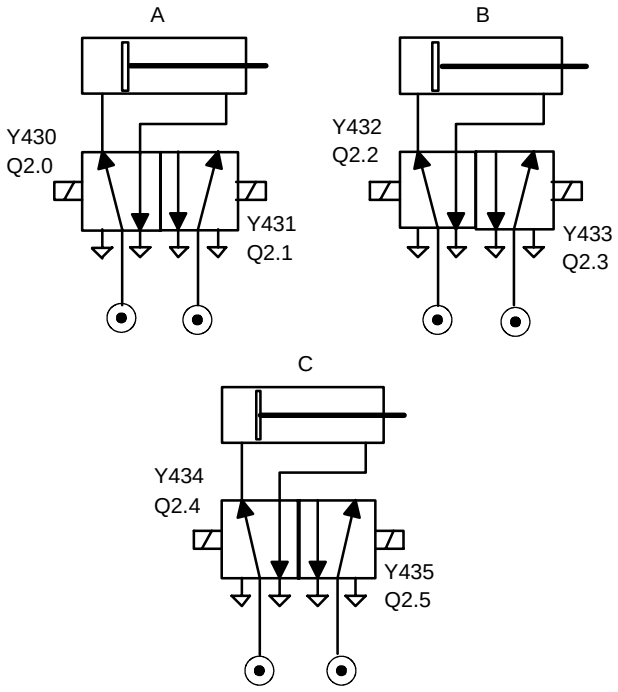


Figure 14.9 *The valves*

X400/I0.0 is the start switch. When it is closed there is an output from Y430/Q2.0 and the timer T450/T0 starts. The start switch is latched by the output. Piston A moves to the right. After the set time, K = 4, the normally closed timer T450/internal relay F0.0 contacts open and the normally open timer T450/internal relay F0.0 contacts close. This switches off Y430/Q2.0 and energises Y431/Q2.1 and starts timer T451/T1. Piston A moves to the left. In rung 2, the T450/internal relay F0.0 contacts are latched and so the output remains on until the set time has been reached. When this occurs the normally closed timer T451/internal relay F0.1 contacts open and the normally open T451/ internal relay F0.1 contacts close. This switches off Y431/Q2.1 and energises Y432/Q2.2 and starts timer T452/T2. Piston B moves to the right. Each succeeding rung activates the next solenoid. Thus in sequence, each of the outputs is energised.

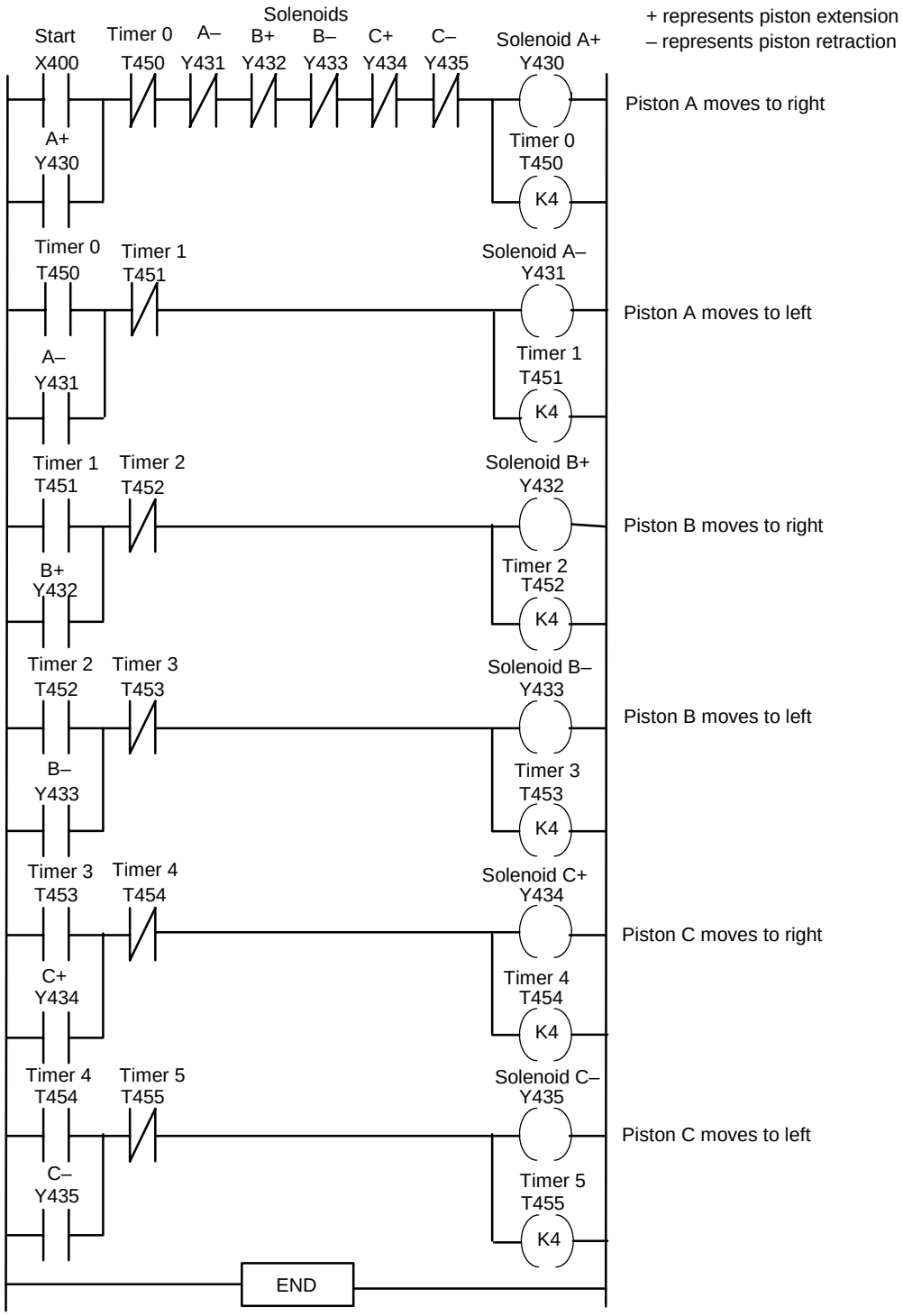


Figure 14.10 *Mitsubishi format program*

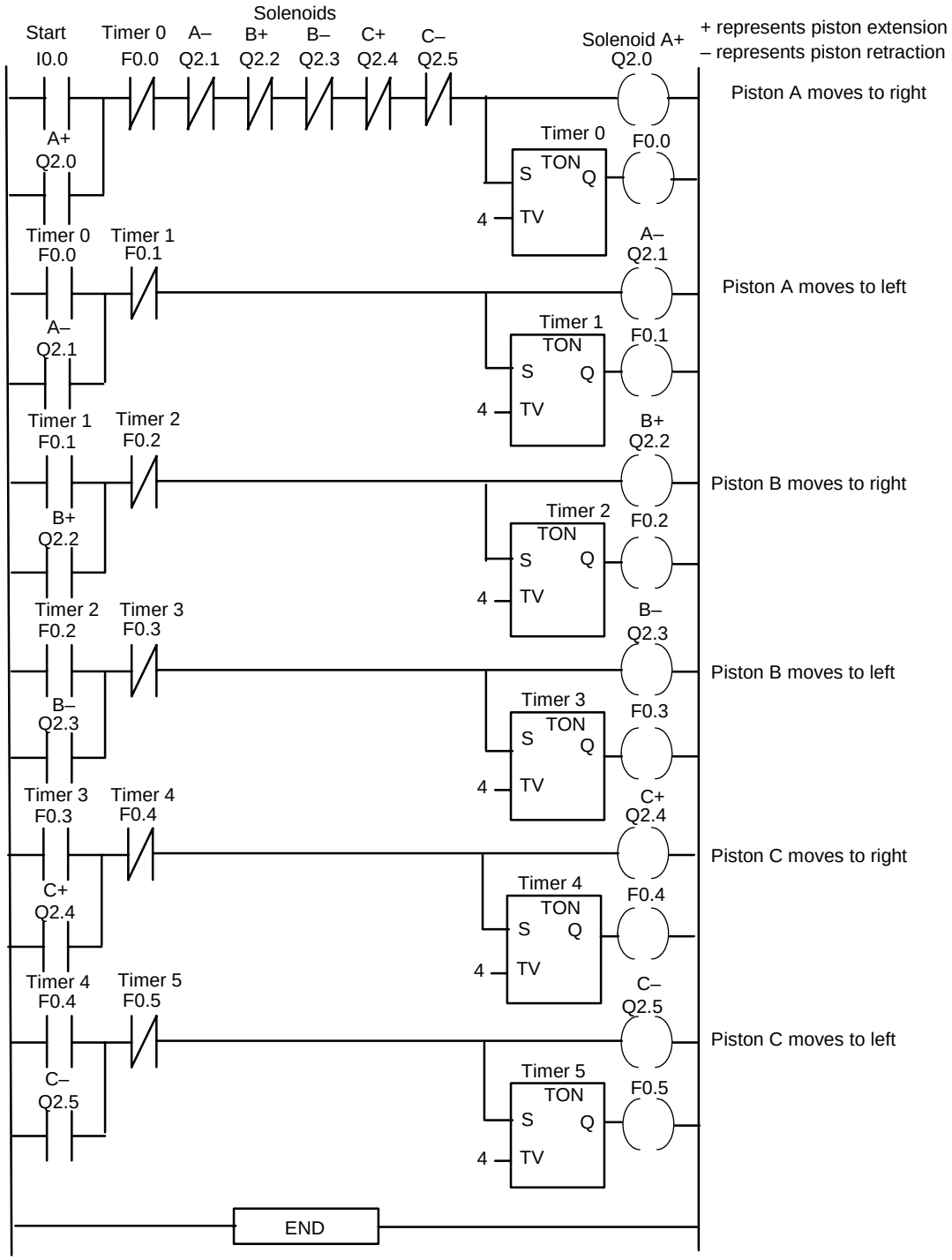


Figure 14.11 *Siemens format program*

The program instruction list, in the Mitsubishi format, for the above program is:

```
LD      X400    (*Start switch*)
OR      Y430
ANI     T450
ANI     Y431
ANI     Y432
ANI     Y433
ANI     Y434
ANI     Y435
OUT     Y430    (*Piston A moves to right*)
OUT     T450    (*Timer T450 starts*)
LD      T450
OR      Y431
ANI     T451
OUT     Y431    (*Piston A moves to left*)
OUT     T451    (*Timer T451 starts*)
LD      T451
OR      Y432
ANI     T452
OUT     Y432    (*Piston B moves to right*)
OUT     T452    (*Timer T452 starts*)
LD      T452
OR      Y433
ANI     T453
OUT     Y433    (*Piston B moves to left*)
OUT     T453    (*Timer T453 starts*)
LD      T453
OR      Y434
ANI     T454
OUT     Y434    (*Piston C moves to right*)
OUT     T454    (*Timer T454 starts*)
LD      T454
OR      Y435
ANI     T455
OUT     Y435    (*Piston C moves to left*)
OUT     T455    (*Timer T455 starts*)
END
```

14.2.2 Sequencing using a sequential function chart

As an illustration of the use of a sequential function chart to describe a program involving sequential control of pneumatic valves and cylinders, consider the situation where we have two cylinders with the required piston sequence being A+. A–, A+, B+ and then simultaneously A– and B–, i.e. piston A moves out to full stroke, then it retracts, then it is switched on again to full stroke, then B is switched on to full stroke and the simultaneously both A and B retract (Figure 14.12(a)). The sequential function chart for the program is shown in Figure 14.12(b)).

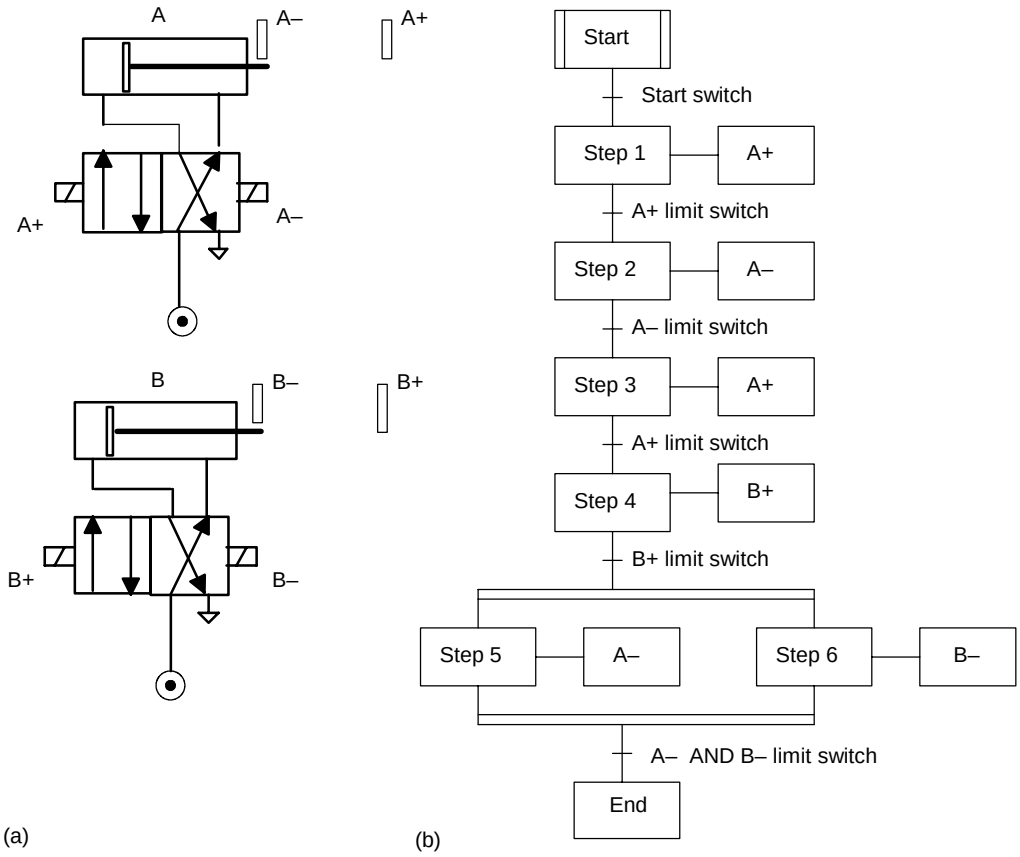


Figure 14.12 *Piston sequence B+. B–, B+, A+ and then simultaneously A– and B–.*

14.2.3 Car park barrier operation using valves

Consider the use of pneumatic valves to operate car park barriers. The in-barrier is to be opened when the correct money is inserted in the collection box, the out-barrier is to open when a car is detected at that barrier. Figure 14.13 shows the type of system that might be used. The valves used to operate the barriers have a solenoid to obtain one position and a return spring to give the second position. Thus when the solenoid is not energised, the position given is that obtained by the spring. The valves are used to cause the pistons to move. When the pistons move upwards the movement causes the barrier to rotate about its pivot and so lift. When a piston retracts, under the action of the return spring, the barrier is lowered. When a barrier is down it trips a switch and when up it trips a switch, these switches being used to give inputs indicating when the barrier is down and up. Sensors are used to indicate when the correct money has been inserted in the collection box for a vehicle to enter and to sense when a vehicle has approached the exit barrier.

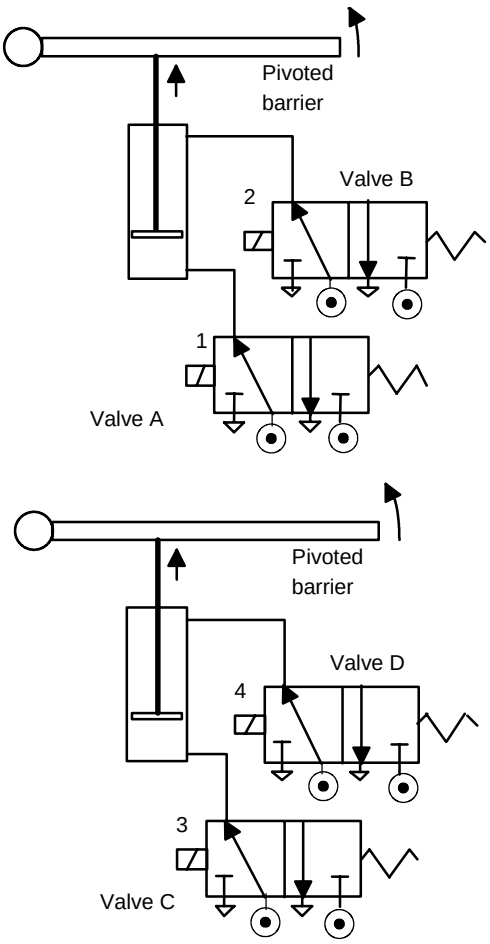


Figure 14.13 *Valve-piston system*

Figure 14.14 shows the form a ladder program could take, (a) being the Mitsubishi program and (b) the Siemens program. The output Y430/Q2.0 to solenoid 1 to raise the entrance barrier is given when the output from the coin box sensor gives the X400/I0.0 input. The Y430/Q2.0 is latched and remains on until the internal relay M100/F0.1 opens. The output will also not occur if the barrier is in the process of being lowered and there is the output Y431/Q2.1 to solenoid 2. The timer T450/T1 is used to hold the barrier up for 10 s, being started by input X402/I0.2 from a sensor indicating the barrier is up. At the end of that time, the output Y431/Q2.1 is switched on, activates solenoid 2 and lowers the barrier. The exit barrier is raised by the output Y432/Q2.2 to solenoid 3 when a sensor detects a car and gives the input X401/I0.1. When the barrier is up a timer T451/T2 is used to hold the barrier up for 10 s, being started by input X404/I0.4 from a sensor indicating the barrier is up. At the end of the time, the output Y433/Q2.3 is switched on, activating solenoid 4 and lowering the barrier.

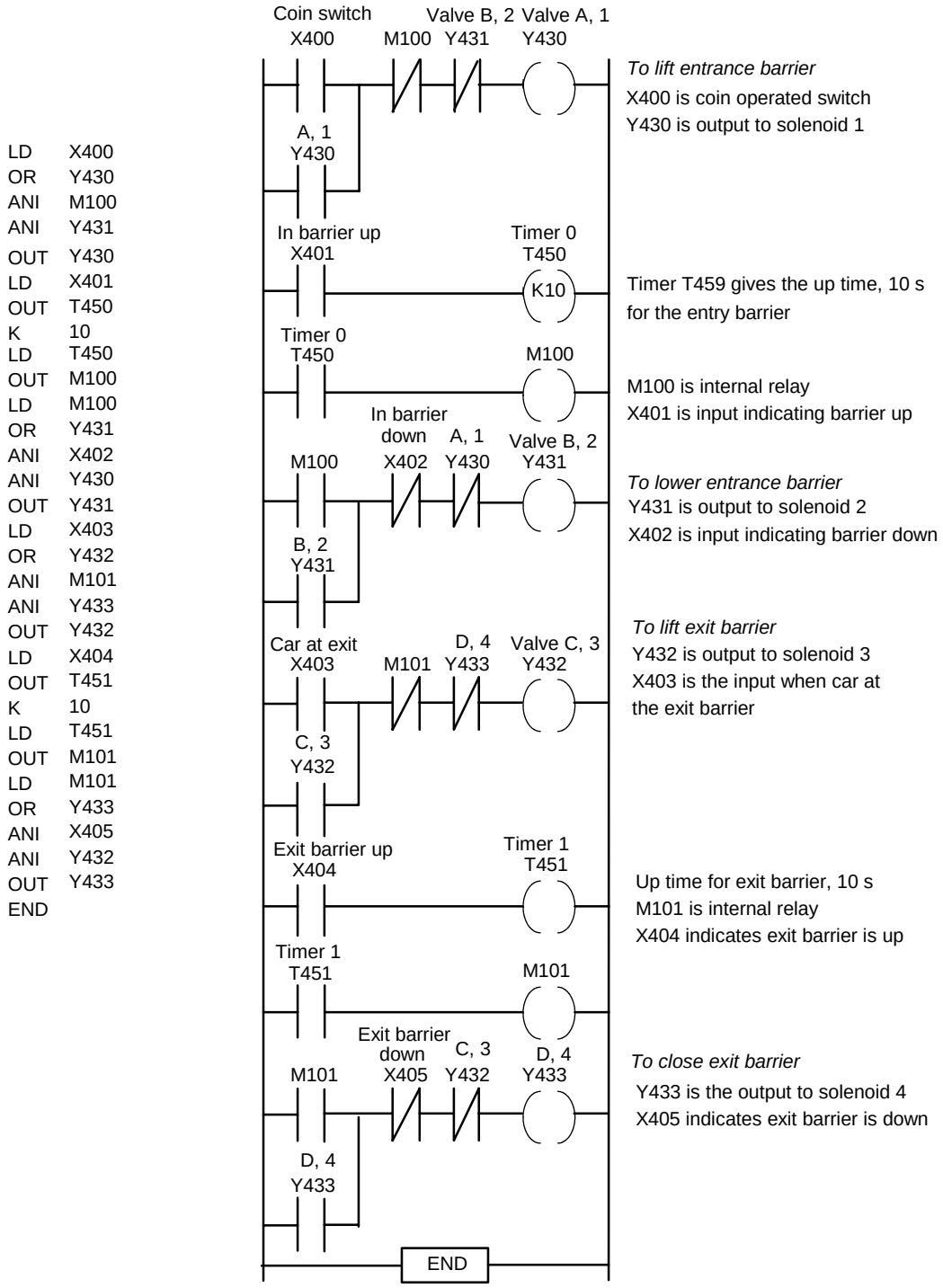
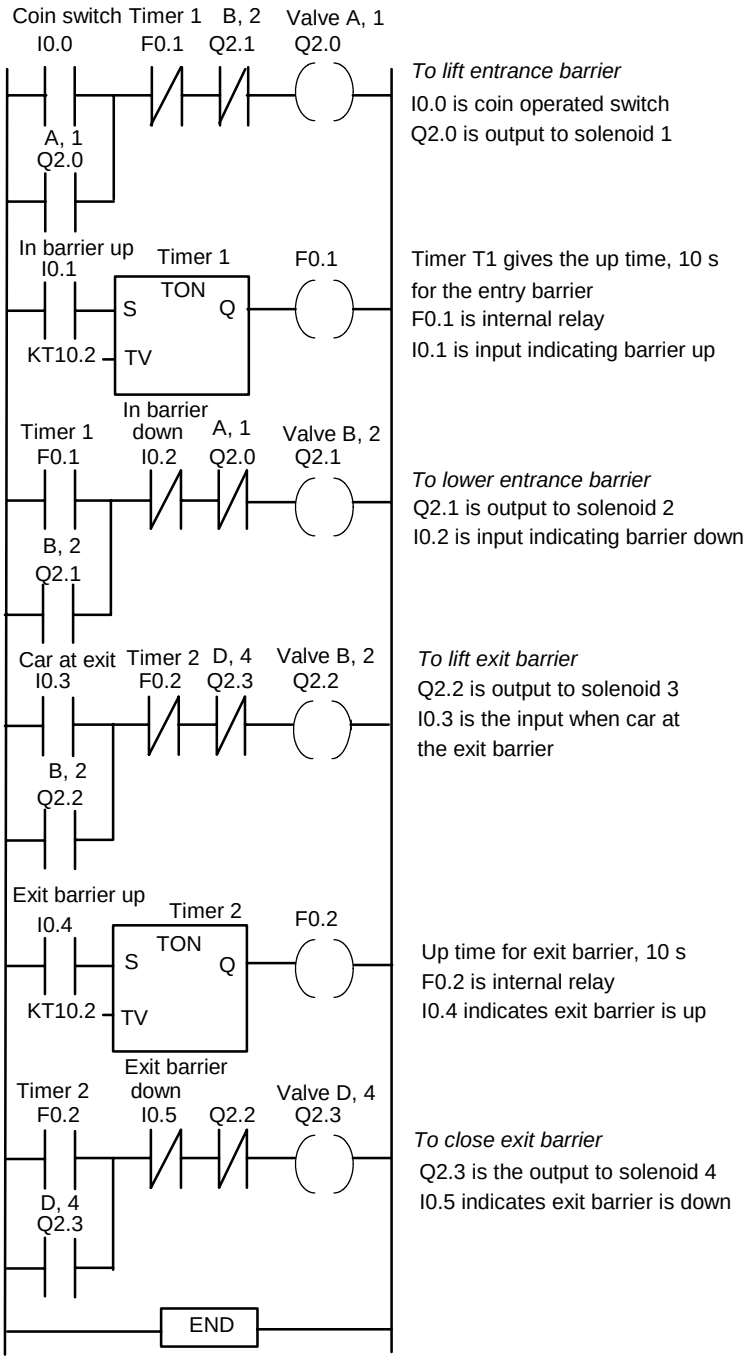


Figure 14.14 *(a) Car barrier program, Mitsubishi format*

```
A     I0.0
O     Q2.0
AN    F0.1
AN    Q2.1
=     Q2.0
A     I0.1
LKT   I0.2
SR    T0
A     T0
=     Q2.0
A     F0.1
O     Q2.1
AN    I0.2
AN    Q2.0
=     Q2.1
A     I0.3
O     Q2.2
AN    F0.2
AN    Q2.3
=     Q2.2
A     I0.4
LKT   I0.2
SR    T1
A     T1
=     F0.2
A     F0.2
O     Q2.3
AN    I0.5
AN    Q2.2
=     Q2.3
END
```



To lift entrance barrier
I0.0 is coin operated switch
Q2.0 is output to solenoid 1

Timer T1 gives the up time, 10 s for the entry barrier
F0.1 is internal relay
I0.1 is input indicating barrier up

To lower entrance barrier
Q2.1 is output to solenoid 2
I0.2 is input indicating barrier down

To lift exit barrier
Q2.2 is output to solenoid 3
I0.3 is the input when car at the exit barrier

Up time for exit barrier, 10 s
F0.2 is internal relay
I0.4 indicates exit barrier is up

To close exit barrier
Q2.3 is the output to solenoid 4
I0.5 indicates exit barrier is down

Figure 14.14 *(b) Car barrier program, Siemens format*

The inputs and outputs for the Mitsubishi program are;

Input		*Output*	
X400	Switch operated by coin	Y430	Valve A, solenoid 1
X401	Input when entrance barrier up	Y431	Valve B, solenoid 2
X402	Input when entrance barrier down	Y432	Valve C, solenoid 3
X403	Input when car at exit barrier	Y433	Valve D, solenoid 4
X404	Input when exit barrier up		
X405	Input when exit barrier down		

and for the Siemens program:

Input		*Output*	
I0.0	Switch operated by coin	Q2.0	Valve A, solenoid 1
I0.1	Input when entrance barrier up	Q2.1	Valve B, solenoid 2
I0.2	Input when entrance barrier down	Q2.2	Valve C, solenoid 3
I0.3	Input when car at exit barrier	Q2.3	Valve D, solenoid 4
I0.4	Input when exit barrier up		
I0.5	Input when exit barrier down		

We could add to this program a system to keep check of the number of vehicles in the car park, illuminating a sign to indicate ‘Spaces’ when the car park is not full and a sign ‘Full’ when there are no more spaces. This could be achieved by using an up and down counter. Figure 14.15 shows a possible Siemens ladder program.

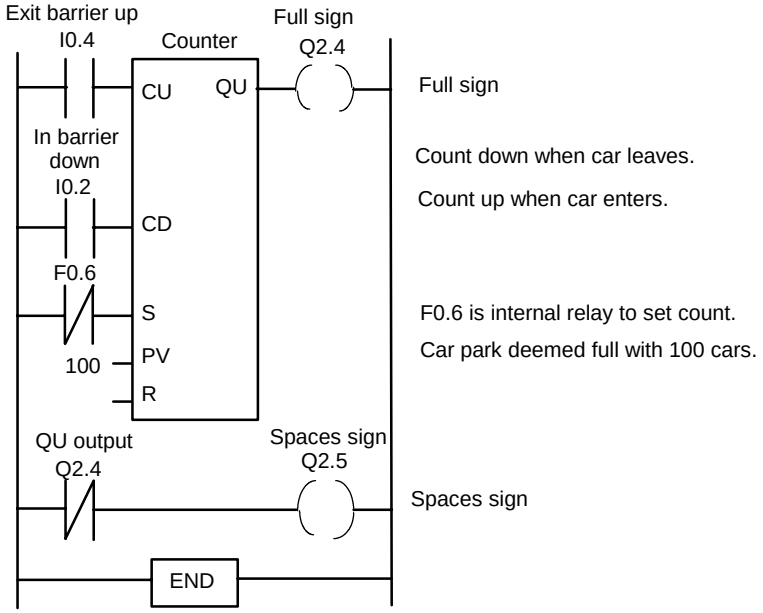


Figure 14.15 *Car park with spaces or full*

14.3 Conveyor belt control

Consider a program that is used to count the number of items put onto a conveyor belt from work cells and give an alert when the number reaches 100. This program might be part of a bigger program used to control a production unit. A proximity sensor can be used to sense when an item is put on the conveyor so that each time a 1 signal is produced. The program might thus be as shown in Figure 14.16, the Allen-Bradley format being used.

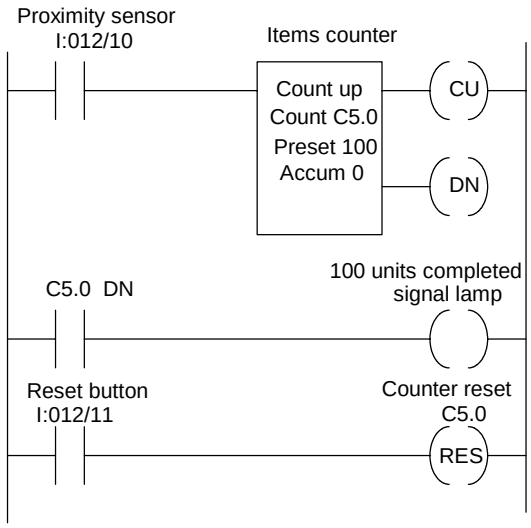


Figure 14.16 *Conveyor belt counting of products*

A further possibility in the above conveyor belt problem is that too many items must not be put on the belt at any one time. A program that might achieve this is to institute a time delay after an item is put on the belt before the next item can be loaded onto the belt. Figure 14.17 shows the program elements for that. When an item passes the proximity sensor, the timer-on-delay is started and only when that is completed will a further item be able to be loaded.

14.3.1 Bottle packing

Consider a production line problem involving a conveyor being used to transport bottles to a packaging unit, the items being loaded onto the conveyor, checked to ensure they are full, capped and then the correct number (4) of bottles being packed in a container. The required control actions are thus: if a bottle is not full the conveyor is stopped; activation of the capping machine when a bottle is at the required position, the conveyor being stopped during this time; count four bottles and activate the packing machine, the conveyor being stopped if another bottle comes to the packing point at that time; sound an alarm when the conveyor is stopped.

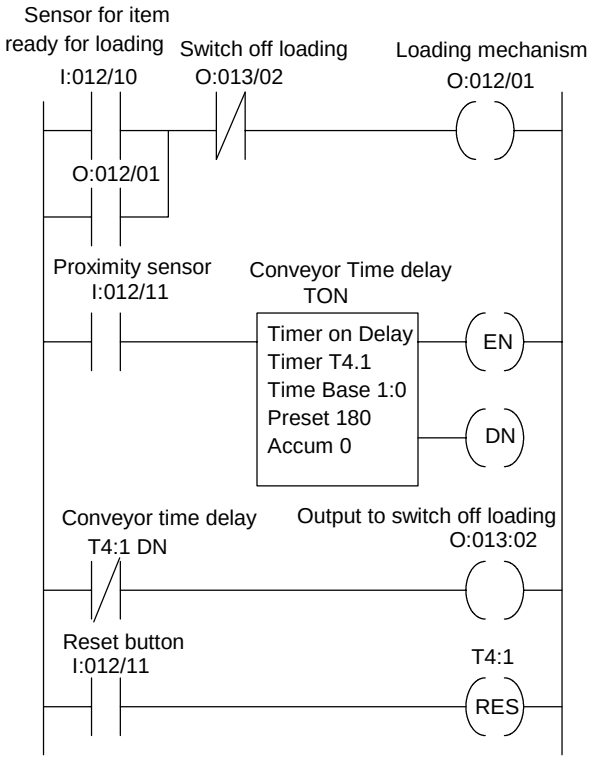
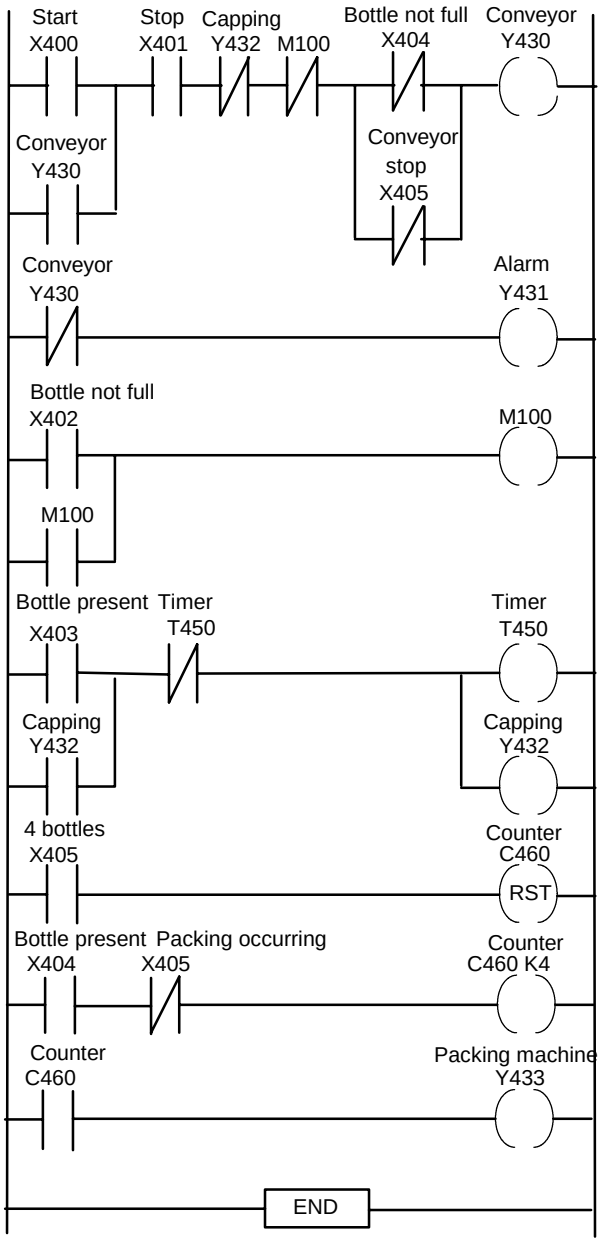


Figure 14.17 *Conveyor belt time delays*

The detection of whether a bottle is full could be done with a photoelectric sensor which could then be used to activate a switch (X402/I0.2 input). The presence of a bottle for the capping machine could also be by means of a photoelectric sensor (X403/I0.3 input). The input to the counter to detect the four bottles could be also from a photoelectric sensor (X404/I0.4 input). The other inputs could be start (X400/I0.0 input) and stop (X401/I0.1 input) switches for the conveyor and a signal (X405/I0.5 input) from the packaging machine as to when it is operating, having got four bottles and so is not ready for any further caps. Figure 14.18 shows a possible ladder program in Mitsubishi format, and Figure 14.19 in Siemens format, that could be used.

The Mitsubishi program in instruction list is:

```
LD      X400        (*First rung*)
OR      Y430
AN      X401
ANI     Y432
ANI     M100
LDI     X404
ORI     X405
ANB
OUT     Y430
LDI     Y430        (*Second rung*)
OUT     Y431
```



Y430 is the output to the conveyor. X400 is the start button, X401 the stop button - externally set closed. The conveyor is stopped by Y232, M100, X404 or X405 being activated.

Y431 is the output to the alarm. It is triggered when the conveyor stops.

M100 is an internal relay activated by X402 closing when a bottle is not full. It then stops the conveyor.

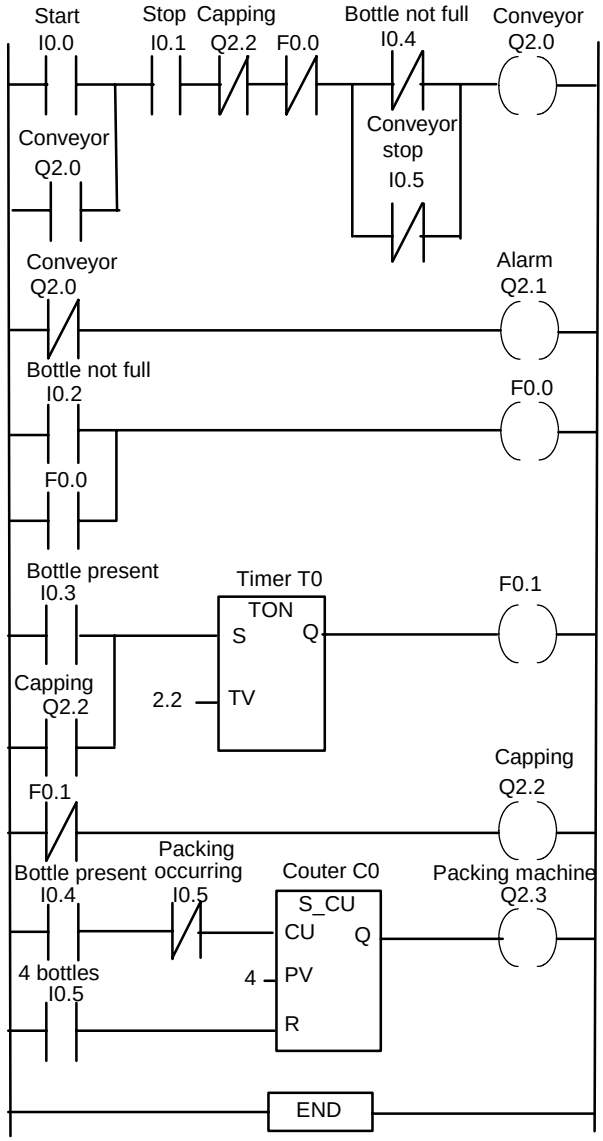
T450 is a timer which stops the conveyor for time taken to cap the bottle. Y432 energises the capping machine and stops the conveyor.

Reset for the counter when packaging machine has 4 bottles.

X404 input when bottle detected. X405 opens when packing occurring. 4 bottles counted.

Y433 energises packing machine when C460 has counted 4 bottles.

Figure 14.18 *Bottle packing program*



Q2.0 is the output to the conveyor. I0.0 is the start button, I0.1 the stop button - externally set. The conveyor is stopped by Q2.2, F0.0, I0.4 or I0.5 being activated.

Q2.1 is the output to the alarm. It is triggered when the conveyor stops.

F0.0 is an internal relay activated by I0.2 closing when a bottle is not full. It then stops the conveyor.

T1 is a timer which stops the conveyor for time taken to cap the bottle.

Q2.2 energises the capping machine and stops the conveyor.

I0.4 input when bottle detected. I0.5 opens when packing occurring. 4 bottles counted.

Q2.3 energises packing machine when counter has counted 4 bottles.

Figure 14.19 *Bottle packing program*

```
LD      X402    (*Third rung*)
OR      M100
OUT     M100
LD      X403    (*Fourth rung*)
OR      Y432
ANI     T450
OUT     T450
K       2       (*2 s allowed for capping.*)
OUT     Y432
LD      X405    (*Fifth rung*)
RST     C460
```

```
LD      X404    (*Sixth rung*)
ANI     X405
OUT     C460
K       4       (*Four bottles counted.*)
LD      C460    (*Seventh rung*)
OUT     Y433
END             (*End rung*)
```

The Siemens program in instruction list is:

```
A       I0.0    (*First rung*)
O       Q2.0
A       I0.1
AN      Q2.2
AN      F0.0
(AN     I0.4
ON      I0.5
)
=       Q2.0
AN      Q2.0    (*Second rung*)
=       Q2.1
A       I0.2    (*Third rung*)
O       F0.0
=       F0.0
A       I0.3    (*Fourth rung*)
O       Q2.2
LKT     2.2     (*2 s allowed for capping.*)
SR      T0
A       T0
=       F0.1
AN      F0.1    (*Fifth rung*)
=       Q2.2
A       I0.4    (*Sixth rung*)
AN      I0.5
CU      C0
LKC     4       (*Four bottles counted.*)
A       I0.5
R       C0
=       Q2.3
END             (*End rung*)
```

14.4 Control of a process

The following is an illustration of the use of a sequential flow chart for programming. The process (Figure 14.20) involves two fluids filling two containers: when full their contents are then emptied into a mixing chamber, from where the mixture is then discharged. The whole process is then repeated.

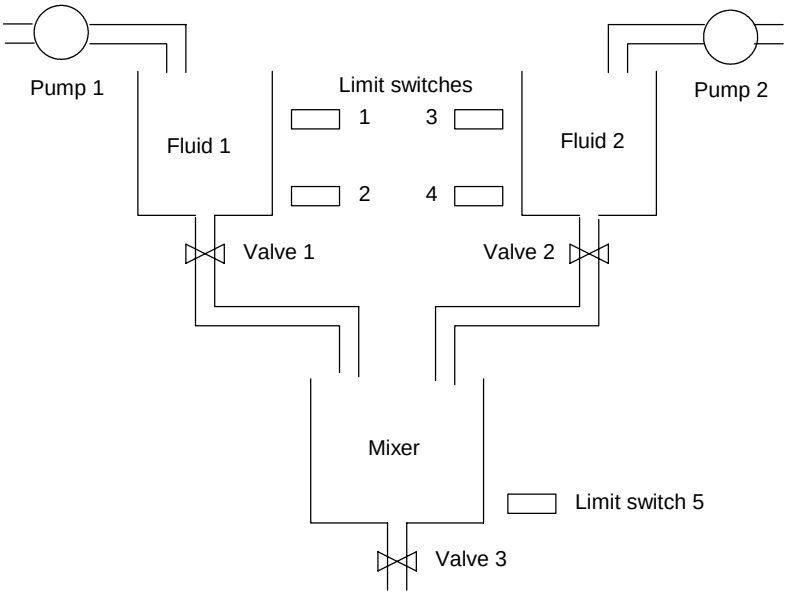


Figure 14.20 *The mixing operation*

Figure 14.21 shows the type of valve that might be used in such a process. It is solenoid operated to give flow through the valve and then, when the solenoid is not activated, a spring returns the valve to the closed position.

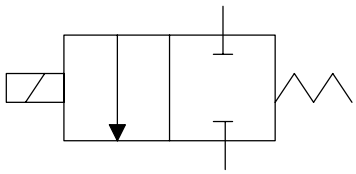


Figure 14.21 *Valve*

Figure 14.22 shows the sequential function chart program. When the start switch is activated, Fill 1 and Fill 2 occur simultaneously as a result of the actions of pumps 1 and 2 being switched on. When limit switch 1 is activated then Fill 1 ceases, likewise when limit switch 3 is activated Fill 2 ceases. We then have the containers for fluid 1 and fluid 2 full. The action that occurs when both limit switch 1 and 3 are activated is that the containers start to empty, the action being the opening of valves 1 and 2. When limit switches 2 and 4 are activated then the containers are empty. The next stage, the mixing of the liquids is then determined when limit switch 2 and limit switch 4 are both activated. After a time of 100 s the moxing ceases and the mixed liquids empty through valve 3. When limit switch 5 is activated the program reaches the end of its cycle and the entire sequence is then repeated.

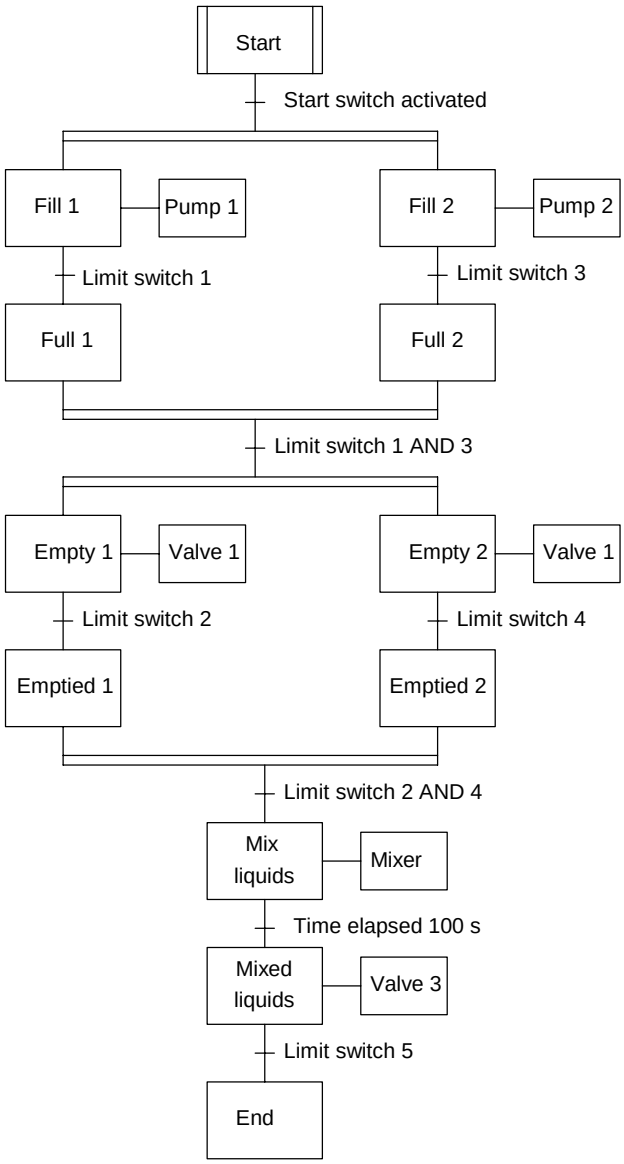


Figure 14.22 *The mixing operation program*

Problems

1 This problem is essentially part of the domestic washing machine program. Devise a ladder program to switch on a pump for 100 s. It is then to be switched off and a heater switched on for 50 s. Then the heater is switched off and another pump is used to empty the water.

2 Devise a ladder program that can be used with a solenoid valve controlled double-acting cylinder, i.e. a cylinder with a piston which can be moved either way by means of solenoids for each of its two positions, and which moves the piston to the right, holds it there for 2 s and then returns it to the left.

3 Devise a ladder program that could be used to operate the simplified task shown in Figure 14.23 for the automatic drilling of workpieces. The drill motor and the pump for the air pressure for the pneumatic valves have to be started. The workpiece has to be clamped. The drill has then to be lowered and drilling started to the required depth. Then the drill has to be retracted and the workpiece unclamped.

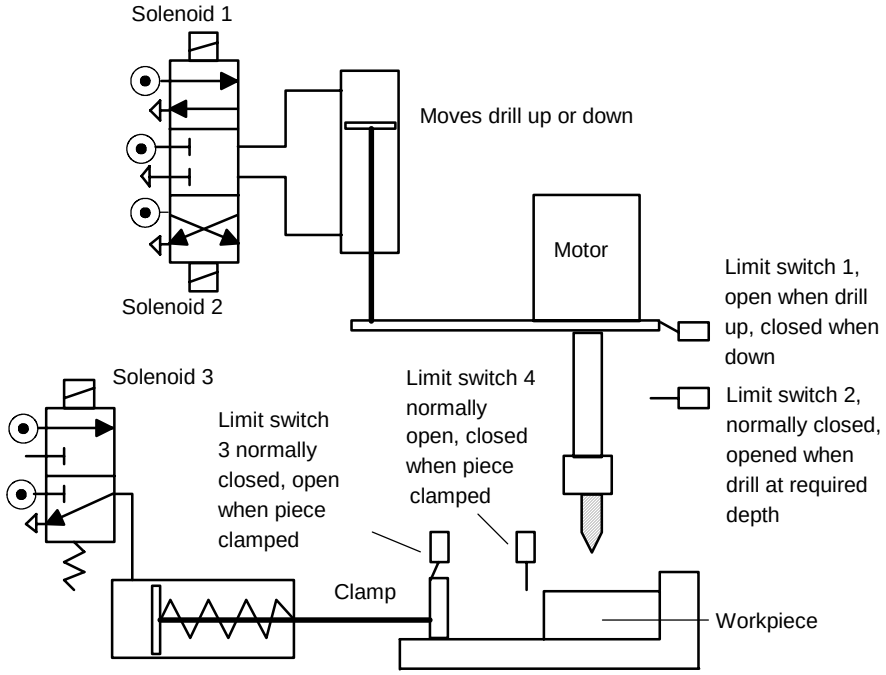

Solenoid 1
Moves drill up or down
Motor
Solenoid 2
Limit switch 1, open when drill up, closed when down
Solenoid 3
Limit switch 4 normally open, closed when piece clamped
Limit switch 2, normally closed, opened when drill at required depth
Limit switch 3 normally closed, open when piece clamped
Clamp
Workpiece

Figure 14.23 *Problem 3*

4 What are the principles to be observed in installing a safe emergency stop system with a PLC?

5 The inputs from the limit switches and the start switch and the outputs to the solenoids of the valves shown in Figure 14.24 are connected to a PLC which has the ladder program shown in Figure 14.25. What is the sequence of the cylinders?

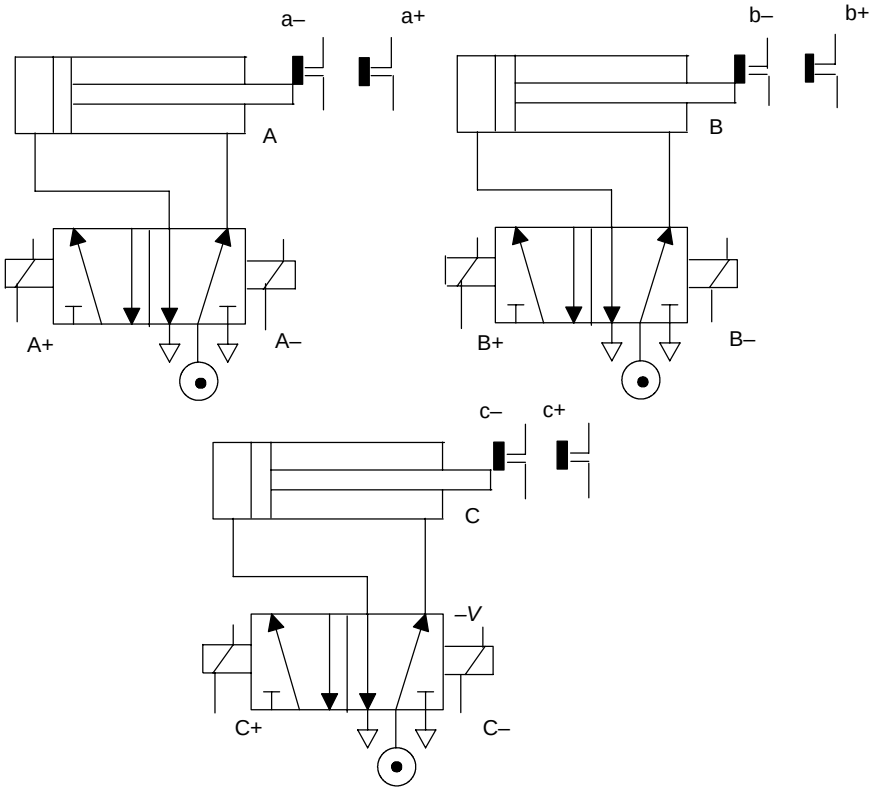


Figure 14.24 *Problem 5*

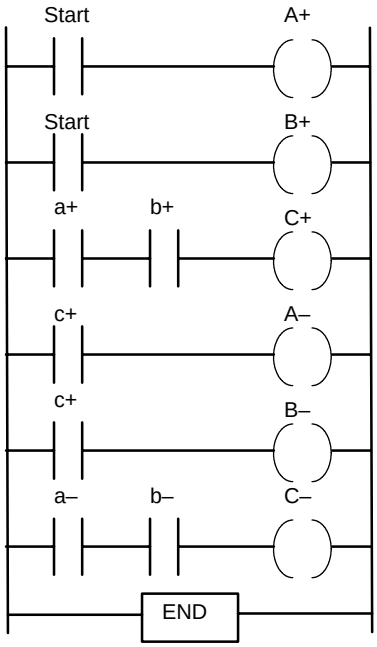


Figure 14.25 *Problem 5*

6 The inputs from the limit switches and the start switch and the outputs to the solenoids of the valves shown in Figure 14.26(a) are connected to a PLC which has the ladder program shown in Figure 14.26(b). What is the sequence of the cylinders?

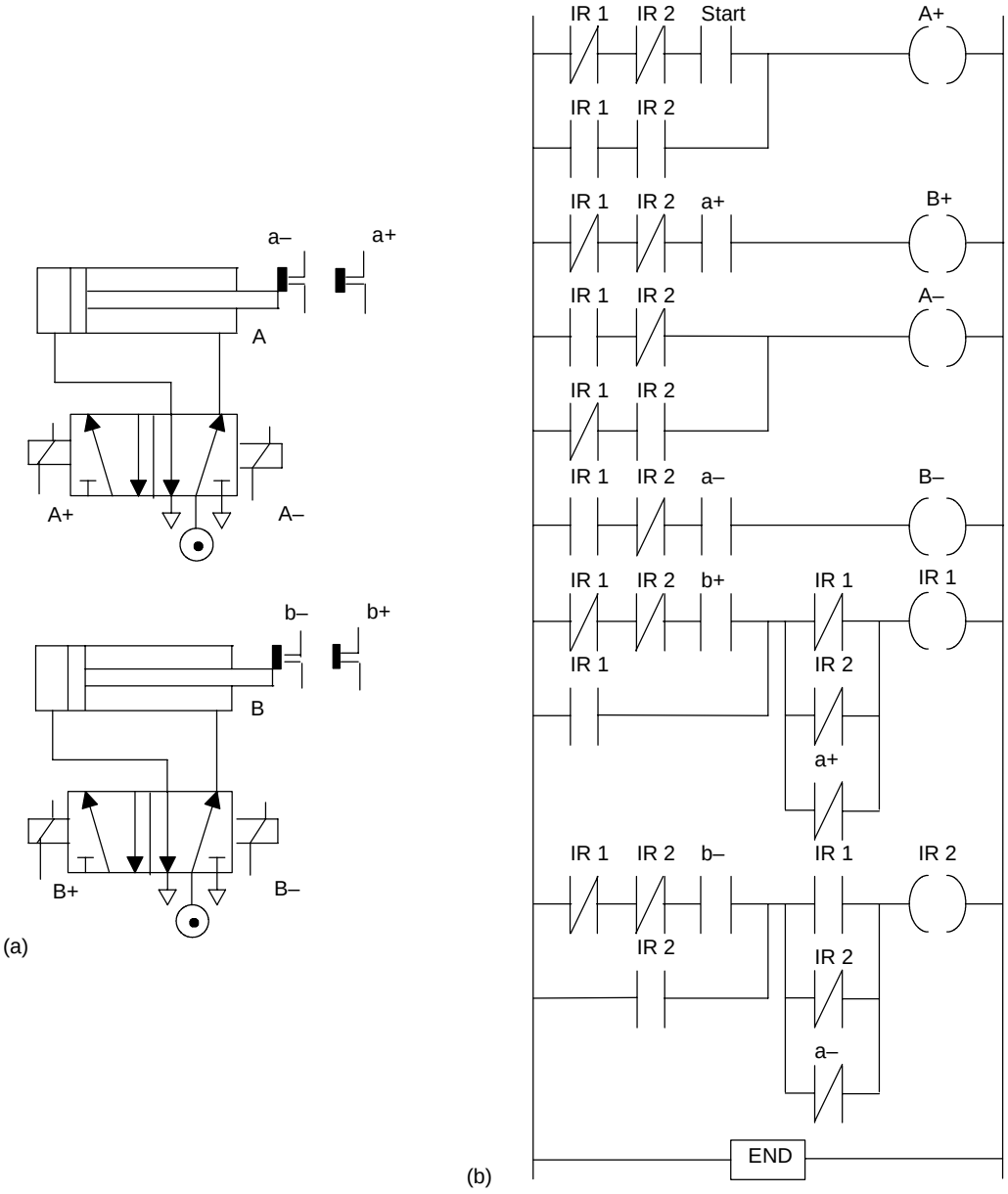


Figure 14.26 *Problem 6*

7 Figure 14.27 shows a ladder program involving a counter C460, inputs X400 and X401, internal relays M100 and M101, and an output Y430. X400 is the start switch. Explain how the output Y430 is switched on.

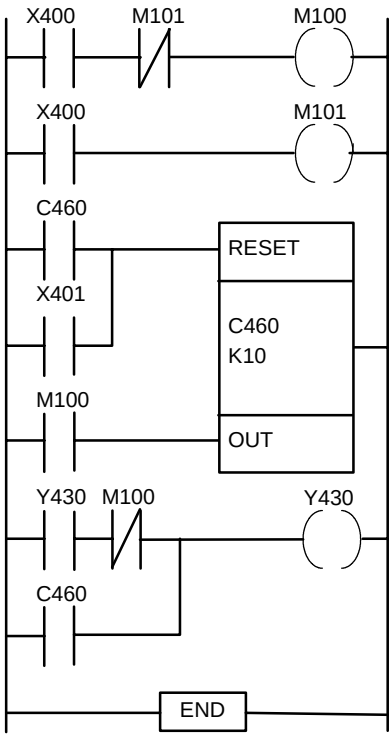
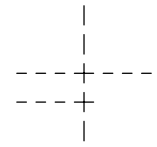
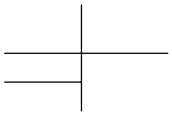
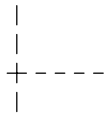
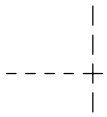
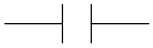
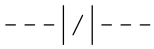
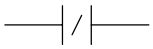
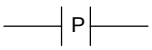
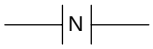
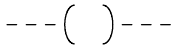
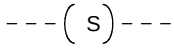
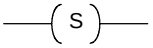
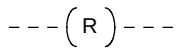
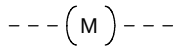
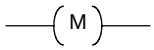


Figure 14.27 *Problem 7*

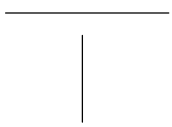
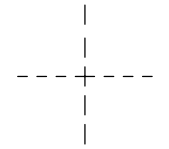
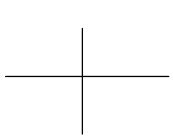
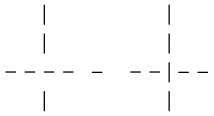
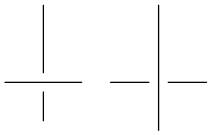
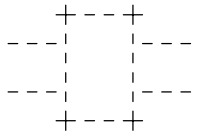
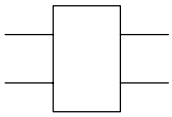
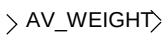
Appendix: Symbols

The following are the main symbols encountered.

Ladder programs

	Semi-graphic form	*Full graphic form*
A horizontal link along which power can flow		
Interconnection of horizontal and vertical power flows		
Left-hand power connection of a ladder rung		
Right-hand power connection of a ladder rung		
Normally open contact		
Normally closed contact		
Positive transition-sensing contact, power flow occurs when associated variable changes from 0 to 1.		
Negative tranistion-sensing contact, power flow occurs when assoaciated variable changes from 1 to 0		
Output coil: if the power flow to it is on then the coil state is on		
Set coil		
Reset coil		
Retentive memory coil, the state of the associated variable is retained on PLC power fail		

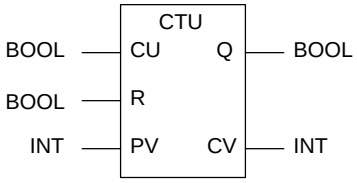
Function blocks

	Semi-graphic form	*Full graphic form*
Horizontal and vertical lines		
Interconnection of horizontal and vertical signal flows		
Crossing horizontal and vertical signal flows		
Blocks with connections		
Connectors		
		

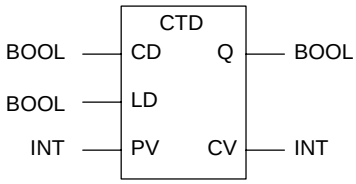
Commonly encountered blocks:

BOOL is Boolean signal, INT is integer, REAL is a floating point number, ANY is any form of signal

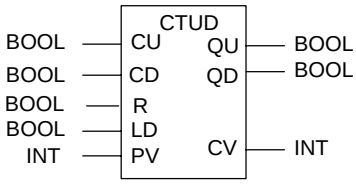
Up-counter counts the number of rising edges at input CU. PV defines the maximum value of the counter. Each new rising edge at CU increments CV by 1. Output Q occurs after set count. R is the reset.



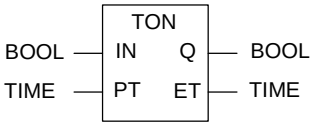
Down-counter counts down the number of rising edges at input CU. PV defines the starting value of the counter. Each new rising edge at CU decrements CV by 1. Output Q occurs when count reaches. zero.



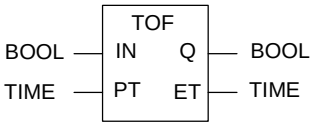
Up-down counter. It can be used to count up on one input and down on the other.



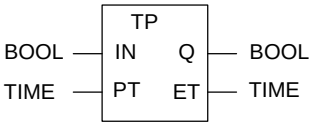
On-delay timer. When input IN goes true, the elapsed time at about ET starts to increase and when it reaches the set time, specified by input PT, the output Q goes true.



Off-delay timer. When input IN goes true, the output Q follows and remains true for the set time after the input Q goes false.

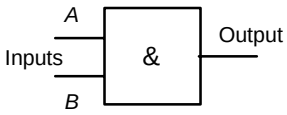


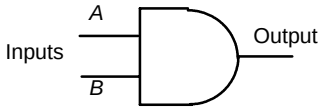
Pulse timer. When input IN goes true, output Q follows and remains true for the pulse duration specified by input PT.



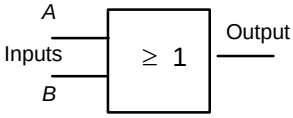
Logic gates

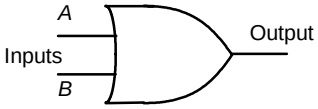
AND gate



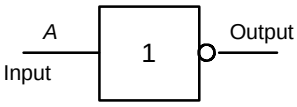


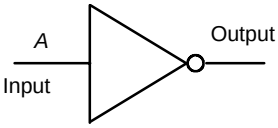
OR gate



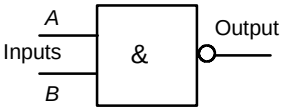


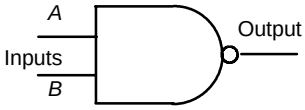
Not gate



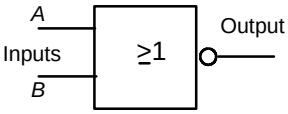


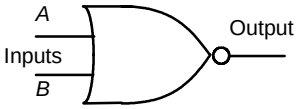
NAND gate



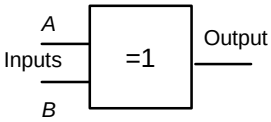


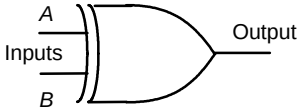
NOR Gate





XOR Gate



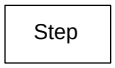


Sequential function charts

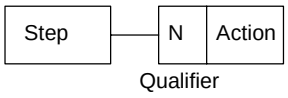
Start step. This defines the step which will be activated when the PLC is Cold-started.

Transition condition. Every transition must have a condition. One that always occurs should be shown with the condition TRUE.

Step in a program

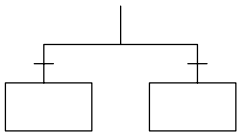


Every step can have an associated action. An action describes the bahviour that occurs when the step is activated. Each action can have a qualifier: N indicates the action is executed while the step is active. If no qualifier is indicated it is taken to be N.

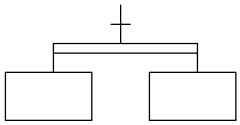


D: time delayed action which starts after a given time.

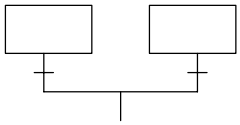
Selective branching



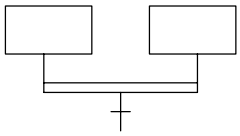
Parallel branching when the transition occurs



Convergence when both transitions occur



Simultaneous convergence



Instruction List (IEC 1131-3 symbols)

LD	Start a rung with an open contact.
LDN	Start a rung with a closed contact
ST	An output
S	Set true
R	Reset false
AND	Boolean AND
ANDN	Booleand NAND
OR	Boolean OR
ORN	Boolean NOR

XOR	Boolean XOR
NOT	Boolean NOT
ADD	Addition
SUB	Subtraction
MUL	Multiplication
DIV	Division

Structured text

X := Y	Y represents an expression that produces a new value for the variable X.

Operators

(...)	Parenthised (bracketed) expression
Function(...)	List of parameters of a function
**	Raising to a power
-, NOT	Negation, Boolean NOT
*, /, MOD	Multiplication, division, modulus operation
+, -	Addition, subtraction
<, >, <=, >=	Less than, greater than, less than or equal to, greater than or equal to
=, <>	Equality, inequality
AND, &	Boolean AND
XOR	Boolean XOR
OR	Boolean OR

Conditional and iteration statements

IF ... THEN ... ELSE is used when selected statements are to be executed when certain conditions occur.

The FOR ... DO iteration statement allows a set of statements to be repeated depending on the value of the iteration integer variable.

The WHILE ... DO iteration statement allows one or more statements to be executed while a particular Boolean expression remains true.

The REPEAT ... UNTIL iteration statement allows one or more statements to be executed and repeated whilst a particular Boolean expression remains true.

Answers

Chapter 1

1 D 2 A 3 C 4 A 5 A 6 C

7 See Figure 1.4.
8 See Figure 1.7 and associated text.
9 See Section 1.3.4.
10 2×1024.

Chapter 2

1 A 2 A 3 B 4 D 5 C 6 A

7 A 8 A 9 B 10 C

11 See (a) Figure 2.5, (b) Section 2.1.4, (c) Section 2.1.3, (d) Section 2.1.8.
12 See Section 2.2.3.
13 See Section 2.2.4.

Chapter 3

1 (a) 3, (b) 63, (c) 13
2 (a) 110 0100, (b) 1001 0010, (c) 1111 1111
3 (a) 159, (b) 3411, (c) 1660
4 (a) E, (b) 51, (c) A02
5 (a) 1110, (b) 11101, (c) 1010 0110 0101
6 (a) 250, (b) 12, (c) 1376
7 (a) 24, (b) 411, (c) 620
8 (a) 010 111 000, (b) 001 000 010, (c) 110 111 011
9 (a) 0010 0000, (b) 0011 0101, (c) 1001 0010
10 (a) 1111 1111, (b) 1101 1101, (c) 1000 0011
11 (a) –16, (b) –55, (c) –40
12 (a) 0.110010×2^{-3}, (b) 0.1100×2^{-4}, (c) $0.1000\ 0100 \times 2^{4}$

Chapter 4

1 B 2 A 3 C 4 C 5 A 6 C

7 A 8 D 9 D

10 (a) 0, (b) 1.
11 To detect message corruption.
12 See Section 4.5.

Chapter 5

1 A 2 D 3 B 4 B 5 B 6 B

7 B 8 D 9 C 10 A 11 A 12 B

13 D 14 C 15 B 16 A 17 C 18 C

19 D 20 C 21 A

22 See (a) Figure 5.8, (b) Figure 5.10, (c) Figure 5.19, (d) Figure 5.10, (e) Figure 5.11, (f) Figure 5.5(a).

Chapter 6

1 C 2 A 3 B 4 D 5 A 6 C

7 A 8 B 9 D 10 A 11 B 12 A

13 B 14 A 15 C 16 A 17 D 18 C

19 A 20 C 21 B

Chapter 7

1 D 2 B 3 C 4 A 5 C 6 C

7 A 8 A 9 C 10 D 11 B 12 B

13 B 14 C 15 A 16 A 17 A 18 B

19 A 20 A 21 B

22 See (a) Figure 7.8, (b) Figures 7.10 or 7.11, (c) Figure 7.26.

Chapter 8

1 C 2 B 3 A 4 B

Chapter 9

1 C 2 A 3 D 4 D 5 D 6 D

7 C 8 C 9 B 10 C 11 A 12 A

13 A 14 B 15 D 16 B 17 A 18 D

19 C

20 See (a) Figure 9.4, (b) Figure 9.10, (c) Figure 9.12.

Chapter 10

1 C 2 A 3 C 4 B 5 B 6 B

7 B 8 D 9 C 10 A 11 A 12 B

13 B 14 C 15 B 16 D 17 C 18 C

19 A

20 See (a) Figures 10.3, (b) Figures 10.7.

Chapter 11

1 D 2 C 3 C 4 D 5 A 6 A

7 C 8 C 9 D

10 (a) As Figure 11.1/11.2 with a constant input to In 1/X400, so entering a 1 at each shift, (b) As in Figure 11.3 but instead of faulty item, hook with an item, and instead of good item, hooks with no items.

Chapter 12

1 C 2 C 3 B 4 B 5 A 6 A

7 B 8 B 9 C

10 Similar to (a) Figure 12.5, (b) Figure 12.6.

Chapter 13

1 B 2 C 3 B 4 C 5 A 6 A

7 See Section 13.3.1.
8 Power failure, supply off, power tripped.
9 Wiring fault, device fault.
10 See Figure A.1.
11 See Figure A.2.

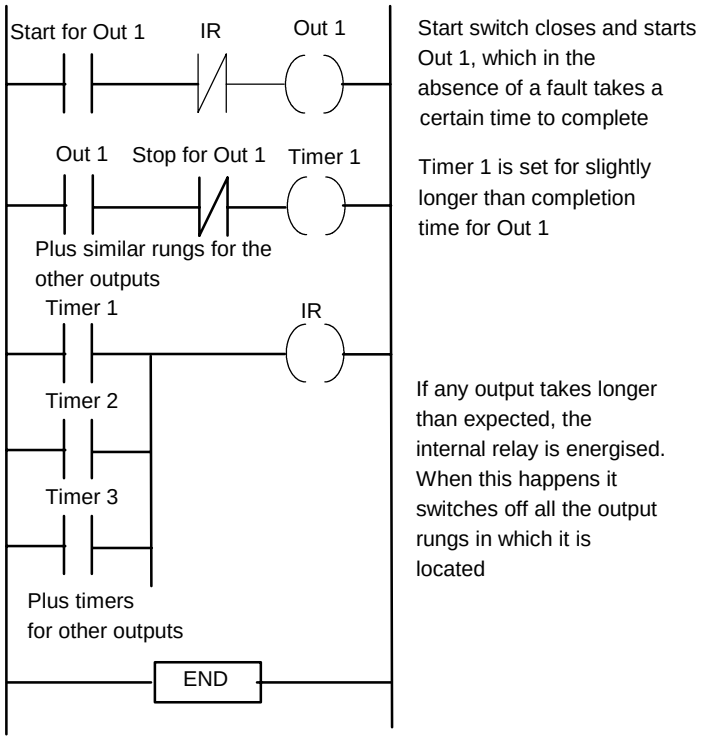
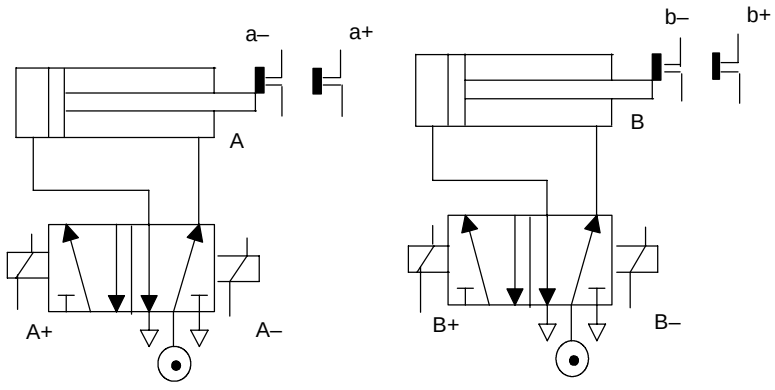
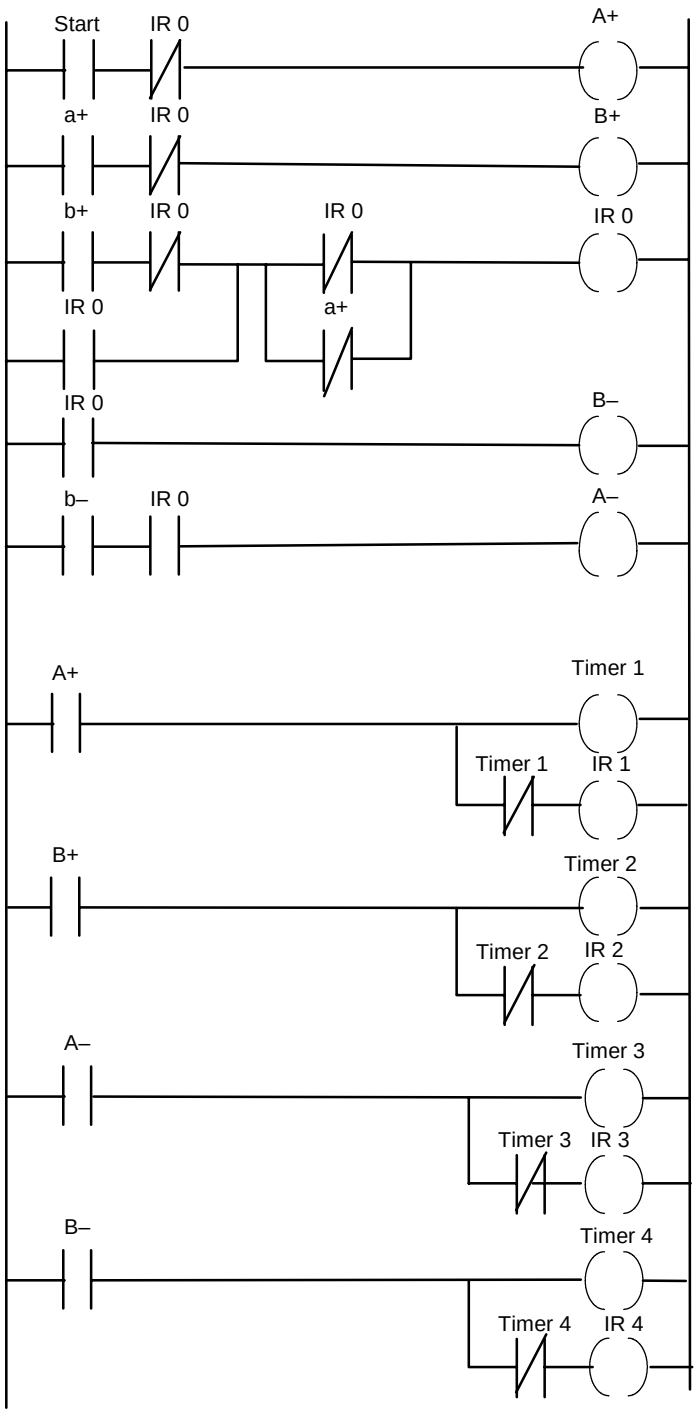

Start for Out 1
IR
Out 1
Start switch closes and starts Out 1, which in the absence of a fault takes a certain time to complete
Out 1
Stop for Out 1
Timer 1
Timer 1 is set for slightly longer than completion time for Out 1
Plus similar rungs for the other outputs
Timer 1
IR
Timer 2
Timer 3
If any output takes longer than expected, the internal relay is energised. When this happens it switches off all the output rungs in which it is located
Plus timers for other outputs
END

Figure A.1 *Chapter 13, problem 10*



(a) The pneumatics

Figure A.2 *(Continued on next page)*



When start switch closed, A+ energised

When a+ activated, B+ energised

When b+ is activated, IR 0 is energised and its contacts close in this rung and later rungs but open in earlier rungs. A+ and B+ switched off

B– switched on

When b– activated, A– energised

The above part of the program gives the sequence A+, B+, B–, A– and the following part the diagnostics

The output A+ produces a short duration pulse at IR 1 as a result of the timer setting

The output B+ produces a short duration pulse at IR 2 as a result of the timer setting

The output A– produces a short duration pulse at IR 4 as a result of the timer setting

The output B– produces a short duration pulse at IR 5 as a result of the timer setting

Figure A.2 *(Continued on next page)*

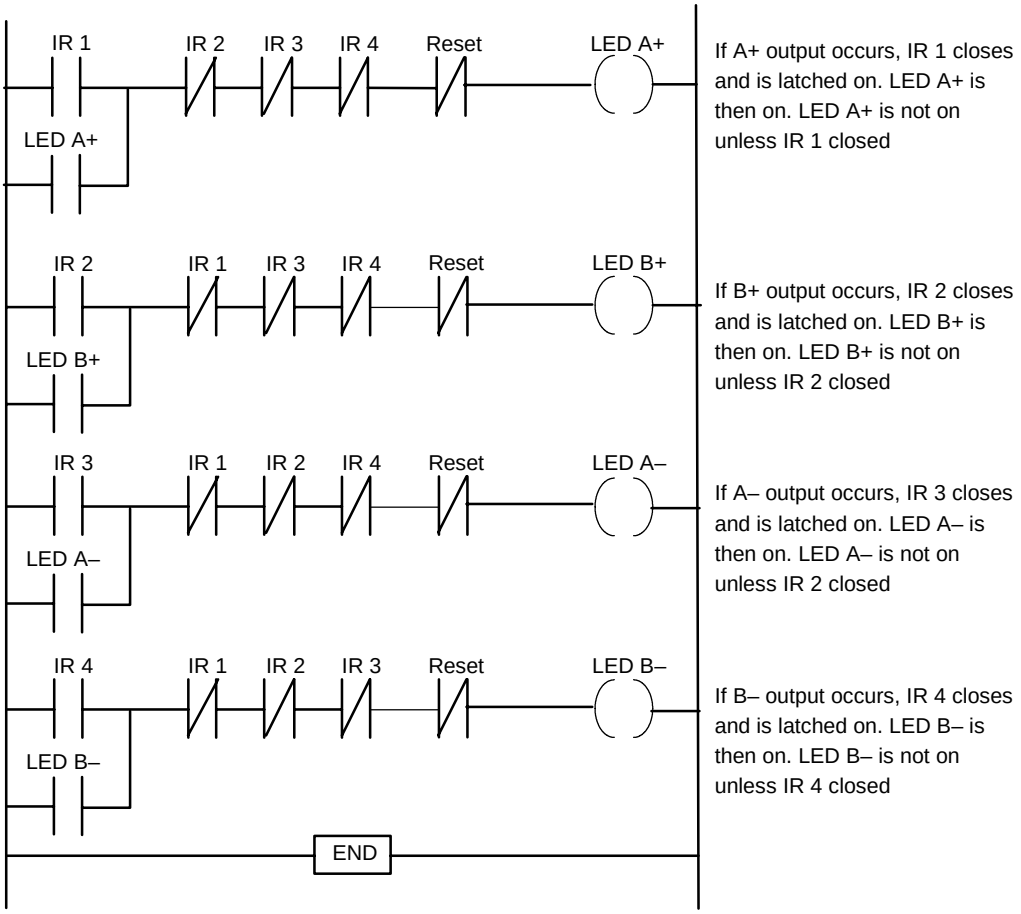

IR 1
IR 2
IR 3
IR 4
Reset
LED A+
LED A+
If A+ output occurs, IR 1 closes and is latched on. LED A+ is then on. LED A+ is not on unless IR 1 closed
IR 2
IR 1
IR 3
IR 4
Reset
LED B+
LED B+
If B+ output occurs, IR 2 closes and is latched on. LED B+ is then on. LED B+ is not on unless IR 2 closed
IR 3
IR 1
IR 2
IR 4
Reset
LED A–
LED A–
If A– output occurs, IR 3 closes and is latched on. LED A– is then on. LED A– is not on unless IR 2 closed
IR 4
IR 1
IR 2
IR 3
Reset
LED B–
LED B–
If B– output occurs, IR 4 closes and is latched on. LED B– is then on. LED B– is not on unless IR 4 closed
END

Figure A.2 *Chapter 13, problem 11*

Chapter 14

1 See Figure A.3.
2 See Figure A.4.
3 See Figure A.5 for a basic answer.
4 Hard-wired emergency stop button, not dependent on software.
5 A+ and B+, C+, A– and B–, C–.
6 A+, B+, A–, B–, A+, A–.
7 M100 and M101 activated. Ten pulses on X401 counted. Then output.

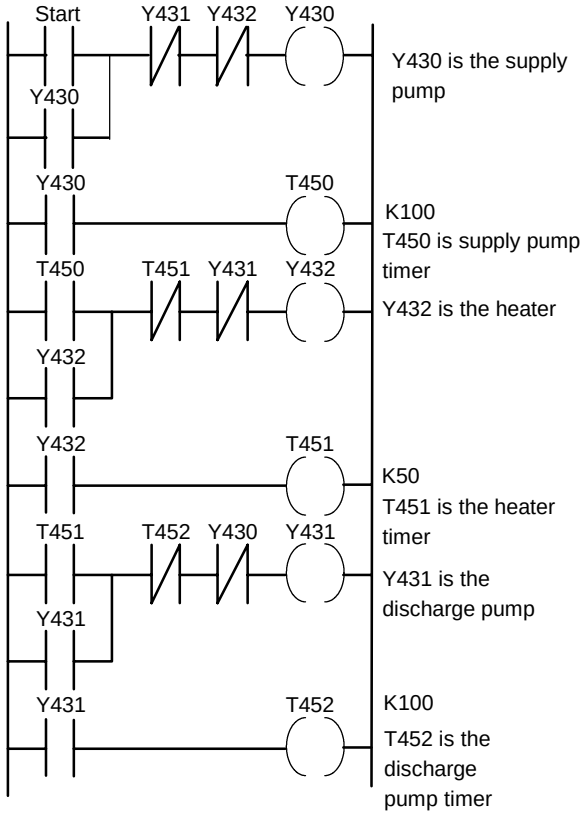


Figure A.3 *Chapter 14, problem 1*

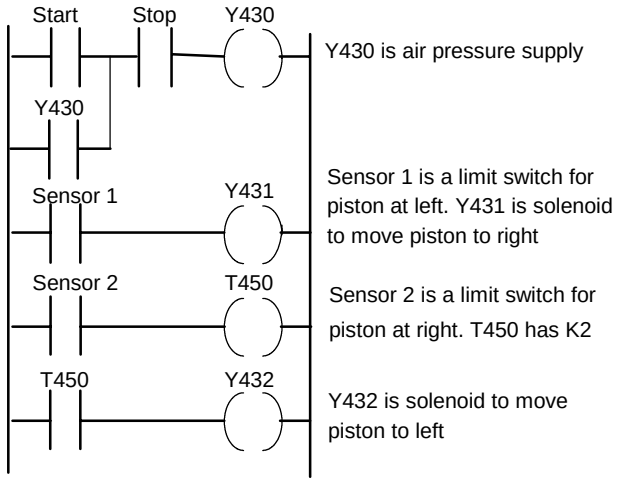

Start
Stop
Y430
Y430 is air pressure supply
Y430
Sensor 1
Y431
Sensor 1 is a limit switch for piston at left. Y431 is solenoid to move piston to right
Sensor 2
T450
Sensor 2 is a limit switch for piston at right. T450 has K2
T450
Y432
Y432 is solenoid to move piston to left

Figure A.4 *Chapter 14, problem 2*

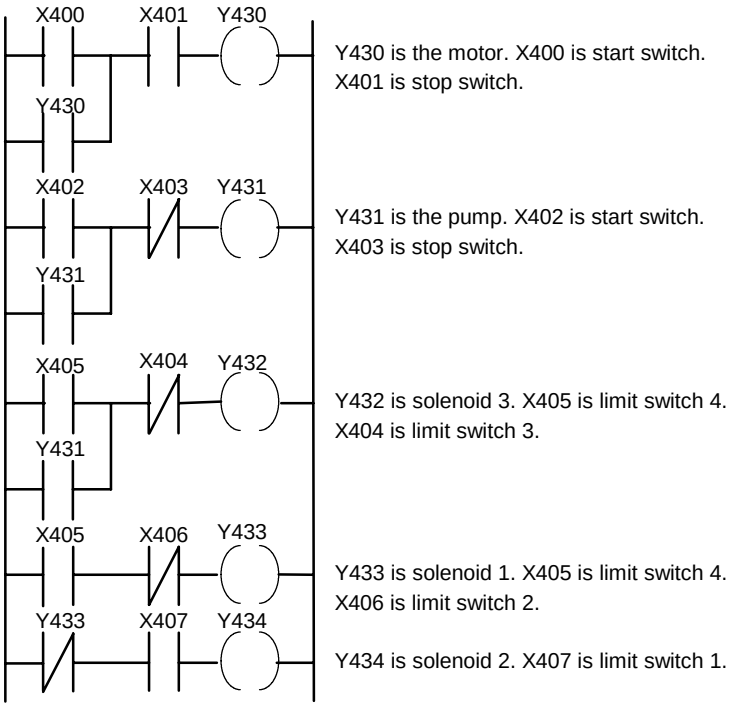


Figure A.5 *Chapter 14, problem 3*

Index

Absolute encoder, 22
Accuracy, 17
Actuator, 30
ADC, 55
Addresses, 76, 82
Alarm system, 141, 201
Algorithm, 210
Allen-Bradley
  addresses, 77
  arithmetic operations, 202
  bit storage, 133
  counter, 174, 178
  data comparison, 200
  data highway, 75
  jump, 154
  latch/unlatch, 136, 139
  master control relay, 143
  move, 198
  PLC-5, 12
  program, 84, 140, 227, 240, 265
  RsLogix, 15, 93
  sequencer, 182
  subroutine, 156
  timer, 160, 164
  timer extension, 179
Amplifier, operational, 60
Analogue
  signals, 5, 53
  standard, 59
  -to-digital converter, 55
AND, 84
Arithmetic and logic unit, 6
ASCII code, 69
Assembly code, 81

Baud rate, 65
BCD, *see* Binary coded decimal system
Bimetal strip, 24
Binary
  arithmetic, 47
  coded decimal system, 47, 197
  number system, 44
  one's complement, 49
  signed, 48
  two's complement, 49, 197
Bits, 6, 44, 189
Bit storage, 132
BOOL, 51
Boolean algebra, 97
Box form of PLC, 10
Brick, 10
Bus, 5, 7
Byte, 8

Car park barrier, 260
Carrier sense multiple access, 71
Central processing unit, 5
Coaxial cabling, 63
Commissioning, 218
Contactor, 30
Continuous updating, 75
Control system
  closed loop, 203
  tasks, 1
  unit, 6
Conveyor belt, 39, 265
Counters
  down-counter, 173
  forms of, 173
  IEC 1131-3, 173, 178
  programming, 174
  up and down, 178
  up-counter, 173
CPU, *see* Central processing unit
CSMA, 71
Cutting band saw program, 227
Cycle, 75, 82
Cyclic redundancy check, 69
Cylinder
  control, 33
  double acting, 33

Cylinder, *continued*
single acting, 32

DAC, 57
Darlington pair, 21
Data Highway Plus, 71
Denary number system, 44
Derivative control, 205
Digital
signals, 5, 53
-to-analogue converter, 57
DINT, 51
Discrete signals, 5, 53
Displacement sensor, 26
Documentation, system, 227
Domestic washing machine, 117

EEPROM, 30
Emergency stop
relays, 217
switches, 102, 216
Encoder, 22
EPROM, 7
Error, 17
Ethernet, 74
Expected value checks, 226

Fail-safe, 215
Fault finding, 220
Feedback loop, 97
Fibre-optic cabling, 63
Fixed point numbers, 50
Flags, 65, 132
Flashing light, 166
Flip-flop, 138, 141
Floating point numbers, 50
Flow charts, 210
Flow meter, 29
Function block
AND, 96
equivalent ladder, 96
feedback loop, 97
NAND, 96
negated input/output, 95
NOR, 96
OR, 96
XOR, 96
programming, 94
symbols, 277

GEM-80 addresses, 77
GENET, 71
Gray code, 23

Hexadecimal number system, 46
Hysteresis error, 17

IEC 1131-3, 51, 80, 83, 94, 108, 113, 136, 137, 138, 159, 173, 192, 205
IEC 61508, 215
IEEE 1451.4, 30
IEEE 488, 66
IEEE/ANSI logic symbols, 95
Increment encoder, 22
Input
devices, 17
/output addresses, 76
/output processing, 53
/output unit, 8
processing, 75
unit, 53
Instruction list
AND, 109
and ladder programs, 109
IEC 1131-3, 108, 113
labels, 109
mnemonics, 108, 279
NAND, 111
NOR, 110
OR, 110
XOR, 112
INT, 51
Integral control, 205
ISO/OSI model, 72

Jump, 154

Ladder programming
AND, 85, 133
and flow charts, 210
and instruction lists, 109
counters, 173
data comparison, 200
entering, 93
equivalent functional block, 97
equivalent sequential function chart, 117, 119
internal relays, 132
latching, 90, 134
monitoring, 221

Ladder programming, *continued*
multiple outputs, 91
NAND, 88
NOR, 89
NOT, 87
OR, 86
principle, 80
sequenced outputs, 92
shift register, 190
symbols, 83, 276
timers, 160
XOR, 90
LAN, 69
Last output set, 222
LED, 21
Lift, 39
Light-emitting diode, 21
Limit switch, 20
Linear variable differential transformer, 26
LINT, 51
Liquid level detector, 29, 40
LM35, 25
LM3911N, 25
Logic functions, 84, 95, 278
Loop, 20 mA, 65
LREAL, 51
LVDT, *see* Linear variable differential transformer

Machine code, 80, 93
Manufacturing automation protocol, 73
MAP, 73
Markers, 132
Mass I/O copying, 75
MelsecNET, 71
MELSOFT, 14, 93
Memory, 7
Mitsubishi
addresses, 77
auxiliary relay, 133
cascaded timers, 161
counter, 173, 174, 176
data comparison, 200
jump, 154
marker, 133
master control relay, 143, 144
MELSEC FX3U, 10, 76
monitoring, 220
move, 198, 199
program, 84, 91, 92, 109, 110, 111, 112, 113, 114, 115, 134, 138, 194, 250, 252, 255, 256, 260, 266
pulse operation, 138
shift register, 190, 191
subroutine, 156
timers, 160
Modular form of PLC, 10
Motor d.c.
basic form, 34
brushless, 35
control, 35
Motor, stepper, 36
Motorola MPX100AP, 28

NAND, 87
Negative transition-sensing coil, 138
Network, local area, 69
Non-linearity error, 17
NOR, 88
NOT, 87

Octal number system, 45
OMRON
CPM1A, 11
CX-One, 15
One-shot, 137
Open systems interconnection model, 72
Operational amplifier, 60
Optoisolators, 8
OR, 85
Orifice flow meter, 29
Output
devices, 30
relay, 9, 57
transistor, 9, 57
triac, 10, 57

Parallel communication
principles, 63
standards, 66
Parity check, 68
Photoconductive cell, 21
Photodiode, 21
Photoelectric sensors, 21
PID control, 205
PLC, *see* Programmable logic controller

Position sensor, 26
Positive transition sensing coil, 138
Potential divider, 60
Potentiometer, 26
Pressure sensors, 28
Process control, 269
Programmable logic controller
  advantages, 3
  architecture, 5
  continuous updating, 75
  cycle, 75
  function, 3
  I/O processing, 53, 75
  input units, 53
  internal relays, 132
  linking, 62
  mass I/O copying, 75
  output units, 56
  programming, 14, 80
  safety, 218
  timers, 159
  types, 10
Proportional control, 204
Protocols, 64, 68
Proximity switches, 20
Pseudocode, 210
Pulse width modulation, 34

Rack form of PLC, 10
RAM, 7
Range, 18
REAL, 51
Reed switch, 21
Registers, 6, 189, 197
Relay
  battery-backed, 136
  internal, 132
  master control, 142
  output device, 30
  retentive, 136
Reliability, 19
Remote connections, 62
Replication, 224
Resistive temperature detector, 24
Response time, 18
Retentive memory coil, 136
Rise time, 18
Robot, 39
ROM, 7
RS232, 64
RS422, 65
RS423, 65
RSLogix, 93
RTD, 24

SAA 1027, 38
Safe systems, 214
SCADA, 72
Scientific notation, 51
Sensitivity, 18
Sensor, 17
Sequencer, 180
Sequential function chart
  actions, 119
  and flow charts, 210
  convergence, 117
  equivalent ladder, 117, 119
  parallel branching, 117
  principles, 116
  process control, 269
  sequencing, 250
  selective branching, 117
  states, 116
  steps, 116
  symbols, 279
  transitions, 116
Serial communication
  principles, 63
  RS232, 64
  standards, 63
Set/reset, 138
Settling time, 18
Shift register
  IEC 1131-3 symbol, 192
  principles, 189
Siemens
  addresses, 77
  arithmetic operations, 203
  BCD/binary, 202
  counter, 173, 174, 178
  data comparison, 200
  flags, 133
  jump, 155
  move, 199
  program, 84, 92, 96, 109, 110, 111, 112, 113, 114, 134, 141, 240, 252, 256, 260, 266
  set/reset, 140, 141
  shift register, 192
  STEP 7, 14, 93

Siemens, *continued*
subroutine, 156
timers, 160, 163, 164
Signal
conditioning, 59
lamp program, 100, 114
Simulation, 220
SINECL, 71
Sinking, 10, 53, 56
Smart sensors, 30
Sourcing, 10, 53, 56
Stability, 19
STEP 7, 93
Stepper motor
action, 36
bipolar, 38
hybrid, 37
permanent magnet, 36
unipolar, 38
variable reluctance, 37
Stop switches, 102, 215
Strain gauges, 27, 61
Structured text
assignment, 120
conditional statements, 122
data comparison, 200
iteration statements, 122
language, 120
move, 198
operators, 121
symbols, 280
traffic lights, 167
Subroutine, 156
Switch sensors, 19

TEDs, 30
Telemecanique, 15, 108, 133, 139, 141, 160
Temperature
control, 250
sensors, 24
Testing, 218
Texas Instrument TSL220, 22
Thermocouple, 25
Thermodiode, 24
Thermotransistor, 24
Time representation, 51
Timers
cascaded, 161
IEC 1131-3, 159
off-delay, 159, 163
on-delay, 159
on-off cycle, 162
pulse, 159, 165
programming, 160
sequencing with, 161
types, 159
TIWAY, 71
Toshiba, 133, 141, 160, 162, 174, 181, 191
Traffic lamp sequence, 115, 167
Transducer, 17
Truth table, 85
Twister-pair cabling, 63

UDINT, 51
UINT, 51
ULINT, 51
USINT, 51

Valve
actuation, 32
control positions, 32
directional control, 31
operation program, 100, 115
ports, 32
sequencing, 145, 176, 193, 254, 256
solenoid operated, 31
spool, 31
symbol, 32

Watchdog timer, 221
Wheatstone bridge, 24, 27, 61
Word, 7, 51, 197

XOR, 89